W. Cramer
G. Kane

Il manuale MC68000

SECONDA EDIZIONE



Il manuale MC68000

W. Cramer
G. Kane

Il manuale MC68000

McGraw-Hill Libri Italia srl

Milano · New York · St. Louis · San Francisco · Oklahoma City
Auckland · Bogotá · Caracas · Hamburg · Lisboa · London · Madrid
Mexico · Montreal · New Delhi · Paris · San Juan · São Paulo
Singapore · Sydney · Tokyo · Toronto

Da un originale Osborne/McGraw-Hill

Titolo originale: *68000 Microprocessor Handbook*

Realizzazione editoriale:
EDIGEO srl, via del Lauro 3, 20121 Milano
Traduzione: Stefano Baldini
Copertina: Marco Mazzucato
Stampa: Litovelox, Trento

ISBN 88-386-0058-9

Printed in Italy
1234567890LTVLET89432109

1ª edizione aprile 1989

Indice

Prefazione

La prima edizione de *Il manuale MC68000* documentava gli sforzi compiuti dalla Motorola nel mondo dei 16 bit. L'architettura base del 68000 (manipolazione di dati e di indirizzi interni a 32 bit) lasciava già presagire che il 68000 sarebbe stato solo il primo di una lunga serie di microprocessori sempre migliori e sempre più completi: a conferma di ciò la Motorola ha già provveduto a far uscire una serie di processori che costituiscono sostanziali miglioramenti rispetto al 68000.
Anche se i nuovi processori cercano di mantenere la compatibilità esiste un certo numero di differenze fra ogni nuovo elemento e il suo predecessore. Dal punto di vista software i microprocessori più recenti posseggono nuove modalità di indirizzamento, nuovi tipi di dati e nuove istruzioni. Dal punto di vista hardware gli ultimi arrivati differiscono per la capacità dei bus, per gli assegnamenti ai piedini, nei segnali, nelle interfacce con le periferiche e nel consumo di energia.
Consci di queste differenze abbiamo deciso che era ora di pubblicarne una versione aggiornata. Ringraziamo la Motorola per averci fornito informazioni dettagliate su ciascuno dei nuovi processori e le facciamo i nostri complimenti per aver sviluppato una famiglia di così geniali microprocessori.

1

Introduzione

La famiglia di microprocessori 68000 rappresenta il contributo della Motorola al mondo dei microprocessori a 16 e a 32 bit. A partire dal primo microprocessore, il 68000, fino ad arrivare all'ultimo della serie, il 68020, la Motorola ha migliorato notevolmente i suoi prodotti fino a diventare un autentico leader in questa tecnologia.
La famiglia 68000 fornisce la base per qualsiasi tipo di applicazione, partendo dalle più semplici fino ad arrivare ad applicazioni in cui sono richieste alta velocità e un throughput elevato, quali ad esempio applicazioni in tempo reale o multiutente. Una cosa molto importante è che i nuovi elementi della serie vengono sviluppati in modo che su di essi possa essere eseguito il codice scritto per elementi precedenti: in questo modo è possibile, a fronte di particolari esigenze dell'utente, trasportare senza alcun problema il codice da processori meno complessi a processori più complessi.

1.1 La famiglia 68000

Ecco un elenco dei componenti la famiglia 68000:

68000 Il 68000 rappresenta la prima apparizione della Motorola nel mondo dei 16 bit: questo microprocessore è composto da 17 registri dati e indirizzi a 32 bit, 14 modalità di indirizzamento, 16 megabyte di spazio indirizzabile, 56 tipi di istruzioni, permette il pipelining e supporta 5 tipi di dati. Esso è completato da un bus dati a 16 bit e da un bus indirizzi a 24 bit.

68008 Questo microprocessore fornisce le stesse possibilità offerte dal 68000, con una migliore organizzazione del package. Esso usa un bus a 8 bit per i dati rendendo molto più semplice la realizzazione della scheda e l'accesso a memorie e a periferiche in grado di supportare bus meno capienti.

68010 Questo microprocessore aggiunge supporti hardware di memoria virtuale e implementa una macchina virtuale, tabelle dei vettori multiple e operazioni di looping sulle istruzioni che assicurano prestazioni elevatissime. Come il 68000, il 68010 usa un bus dati a 16 bit e un bus indirizzi a 24 bit.

68012 Questo microprocessore possiede le stesse caratteristiche del 68010 con la differenza che esso usa un bus indirizzi a 30 bit.

68020 È l'ultimo uscito della serie e fornisce bus dati e indirizzi entrambi a 32 bit (64 gigabyte di spazio indirizzabile), interfaccia per coprocessore (floating-point, ad esempio), sette tipi di dati, 18 modalità di indirizzamento, e memoria cache integrata sul chip.

1.2 Scopo del libro

Questo libro fornisce una panoramica, dal punto di vista funzionale, della famiglia 68000, fornendo informazioni specifiche circa le modalità operative, di indirizzamento e circa i tipi di dati supportati. Oltre a ciò vi fornirà una descrizione molto dettagliata della temporizzazione dei segnali per ognuno dei membri della famiglia.
Dal momento che questo è un libro orientato all'hardware, non ci dilungheremo molto sul set di istruzioni. Per maggiori informazioni circa questo argomento riferitevi a un testo specifico.

1.3 Convenzioni per i segnali e la temporizzazione

I segnali possono essere attivi alti, attivi bassi o avere un significato in entrambi gli stati. Un segnale attivo alto è un segnale in cui lo stato alto provoca l'avvenimento di un determinato evento. Un segnale attivo basso provoca un determinato evento quando il segnale è basso: lo stato alto non provoca alcun effetto. Un segnale che ha due stati attivi provoca due eventi diversi, a seconda che il segnale sia alto o basso: questo segnale non possiede uno stato inattivo.
In questo libro un segnale attivo basso viene identificato da una barra posta sopra il nome del segnale: XX identifica un segnale che produce un effetto quando è nello stato basso. Un segnale attivo alto o un segnale con due stati attivi non ha alcuna barra sopra il proprio nome, ad esempio YY.

Nelle descrizioni "testuali" il termine "attivazione" si riferisce ad un segnale che effettua una transizione dal suo stato inattivo al suo stato attivo indipendentemente dal fatto che il segnale sia alto o basso. Il termine "negazione" si riferisce ad un segnale che effettua una transizione dal suo stato attivo a quello inattivo. Nei diagrammi noi indicheremo le transizioni fisiche e useremo anche le seguenti convenzioni:

1. Un segnale basso equivale a nessuna tensione, mentre un segnale alto comporta la presenza di una tensione.



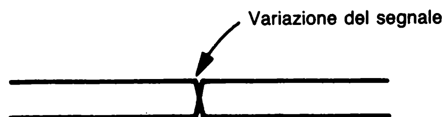
2. Un singolo segnale che effettua una transizione basso-alto viene rappresentato nel modo seguente:



3. Un singolo segnale che effettua una transizione alto-basso viene rappresentato nel modo seguente:



4. Quando esistono due o più segnali paralleli, la notazione:

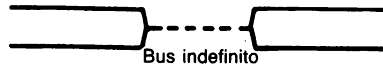


   indica che uno o più dei segnali paralleli cambiano livello, ma non viene indicato il tipo (alto-basso o basso-alto) della transizione.

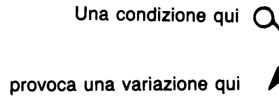
5. Un segnale a tre stati indefinito o fluttuante viene indicato nel modo seguente:



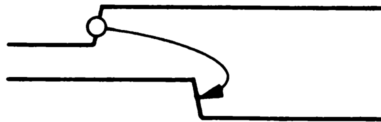
6. Un bus a tre stati indefinito o fluttante viene indicato nel modo seguente:



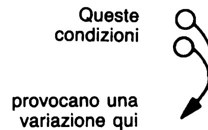
7. Quando la condizione di un segnale provoca il cambiamento di un altro segnale, useremo una freccia per indicare la relazione fra i due segnali:



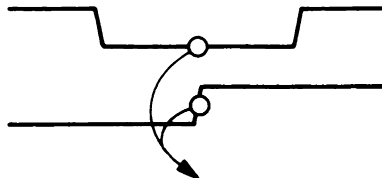
Una transizione di un segnale che influenza un'altra transizione verrà allora illustrata nel modo seguente:



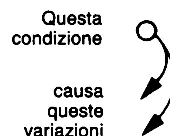
8. Quando sono richieste due condizioni per influenzare un terzo evento, useremo la notazione:



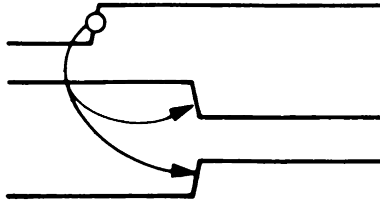
Quando una transizione basso-alto di un segnale accoppiata con una transizione alto-basso di un secondo segnale influenza la transizione di un terzo segnale useremo la notazione:



9. Quando una singola transizione provoca due o più eventi, useremo la notazione:



Quando una transizione basso-alto di un segnale provoca modifiche in altri due segnali, useremo la seguente notazione:



10. Per indicare cambiamenti di livello del segnale useremo onde quadre, ignorando i tempi di caduta e risalita del segnale.

1.4 Nomi dei membri della famiglia

Le similitudini fra i vari membri della famiglia 68000 sono più importanti delle differenze fra di essi. In ogni caso i membri più recenti hanno caratteristiche che non sono presenti nei membri più antichi della famiglia. Descrivendo le proprietà comuni a tutti i membri useremo il termine "68000", mentre quando descriveremo le caratteristiche peculiari di un particolare membro della famiglia specificheremo direttamente il suo nome: ad esempio "68000" si riferisce al primo membro della famiglia "68000".
Notate anche che, mentre nella prima edizione del presente libro usavamo il prefisso "MC" (Motorola Corporation) unitamente al nome del processore (ad esempio "MC 68000"), in questa edizione abbiamo deciso di tralasciare il prefisso. La decisione è stata presa poiché, dal momento che Motorola ha concesso ad altri produttori l'autorizzazione a realizzare dei chip che funzionano in modo perfettamente uguale al Motorola 68000, noi non vogliamo fare alcuna distinzione fra i vari produttori.

2

Descrizione funzionale

Questo capitolo presenta la famiglia 68000 dal punto di vista della programmazione in termini di modalità di esecuzione, registri del processore e modalità di indirizzamento disponibili.

2.1 Modalità di esecuzione

BIT DELLA MODALITÀ SUPERVISORE E UTENTE

Il 68000 può operare in due modalità: supervisore o utente. Il modo in cui opera il processore è stabilito da un bit del suo registro di stato. Come indicato dal nome, la modalità utente è attiva quando il processore esegue programmi applicativi, mentre la modalità supervisore è dedicata a programmi a livello di sistema operativo. La modalità supervisore ha il proprio stack pointer (puntatore allo stack) e ha la possibilità di eseguire istruzioni privilegiate che invece non possono essere eseguite a livello utente.

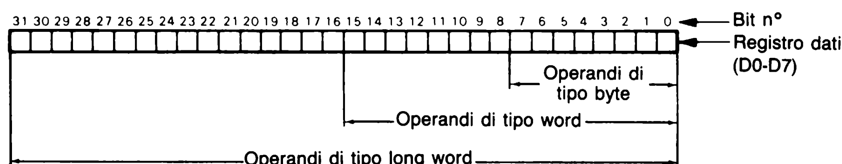
CODICI FUNZIONE

Il chip del processore invia un segnale codificato ai piedini dedicati al codice di funzione (FC0-FC2) che indica tanto la modalità di esecuzione corrente quanto l'attività corrente del bus (accesso ai dati, accesso al programma o segnalazione di interrupt). La logica esterna (ad esempio una unità di gestio-

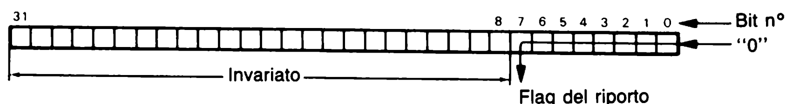
ne della memoria o MMU) può sfruttare queste informazioni per separare la memoria di sistema da quella utente.

2.2 Registri in modalità utente

In modalità utente il 68000 ha otto registri dati a 32 bit (D0-D7), sette registri indirizzi (A0-A6), uno stack pointer a 32 bit (SP), un program counter (contatore di programma) a 32 bit e un registro di stato a 8 bit.
Il registro dati può trattare dati lunghi un bit, dati lunghi 8 bit (1 byte) e dati di 32 bit (long word). Identificheremo il bit meno significativo come bit 0 e il bit più significativo come bit 31. L'illustrazione seguente mostra come gli operandi di varie dimensioni sono posizionati all'interno del registro dati.



Le operazioni a singolo bit comprendono il test, il settaggio e la cancellazione di bit individuali all'interno di un byte, di una parola e di una long word. Le istruzioni per la manipolazione dei bit generalmente forniscono il numero del bit o direttamente o attraverso un registro dati.
Gli operandi di tipo byte occupano i bit da 0 a 7, quelli di tipo word i bit da 0 a 15 e le long word usano il registro intero, ovverosia i bit da 0 a 31. Quando un'istruzione usa un registro dati come operando sorgente o come operando destinazione, verrà alterata solo la parte appropriata del registro, mentre i bit di ordine più elevato del registro rimarranno invariati. Ad esempio un'istruzione di shift aritmetico di un byte a sinistra (ASL.B) shifta solo i bit da 0 a 7 come mostrato dall'illustrazione seguente. I bit da 8 a 31 rimangono dove sono.



Oltre alle loro funzioni di operando sorgente o di operando destinazione, i registri dati possono funzionare anche come registri indice o come contatori dei loop.
Il 68020 permette di usare i dati di tipo quad word, cioè parole lunghe 64 bit, usate solo per istruzioni di moltiplicazione e divisione di dati di 32 bit

(memorizzando il prodotto di due long word o tenendo il dividendo di un divisore di 32 bit). Le quad word possono risiedere in uno dei due registri dati senza restrizioni sull'ordinamento o sull'accoppiamento; i formati delle istruzioni di moltiplicazione e divisione specificano sia i registri alti (i bit da 32 a 63) sia i bit bassi (da 0 a 31).
Il 68020 permette anche di effettuare operazioni sui singoli bit, dando la possibilità di accedere a campi di 32 bit consecutivi, senza richiedere che questi campi debbano corrispondere per forza a indirizzi di byte, word o long word

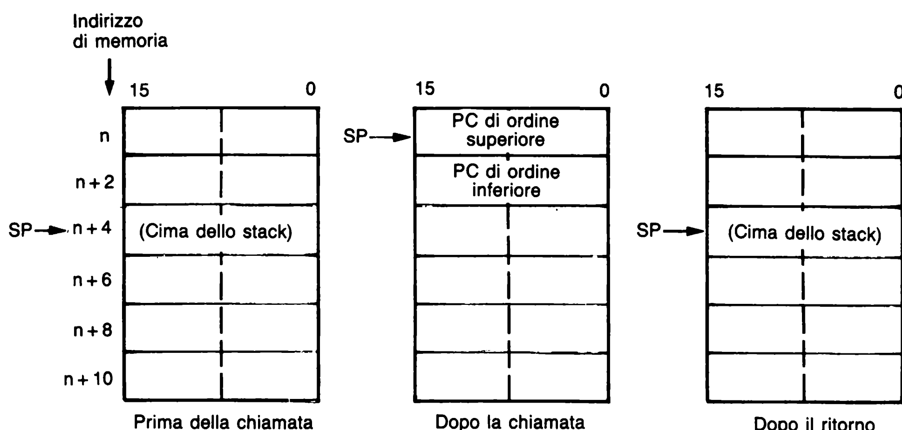
REGISTRI INDIRIZZO

Il 68000 possiede sette registri indirizzo general-purpose: sono i registri A0-A6. A seconda della modalità di indirizzamento questi registri possono contenere indirizzi di operandi, indirizzi di puntatori a operandi, indirizzi base e indici. I registri indirizzo possono contenere dati di 16 o di 32 bit; non vi è alcuna operazione diretta a livello di byte nei registri indirizzo. Quando vengono usati direttamente come operandi sorgente, il segno dei valori a 16 bit viene esteso prima dell'uso (la word di ordine più elevato non ha alcun effetto e non viene toccata dall'operazione). Quando vengono usati direttamente come operandi destinazione i sorgente a 16 bit vengono portati ai 32 bit del registro degli indirizzi destinazione estendendo il segno.

STACK

Il processore usa un ottavo registro indirizzo, A7, come puntatore allo stack. Come già stabilito in precedenza il processore possiede due stack pointer distinti a seconda della modalità di operazione corrente. Questi registri (A7 e A7') sono più comunemente chiamati *user stack pointer* (puntatore allo stack utente), abbreviato con USP, e *system stack pointer* (puntatore allo stack di sistema), abbreviato con SSP. Il processore, basandosi sulla modalità di esecuzione corrente, sa implicitamente quale stack pointer usare per memorizzare le operazioni eseguite. Come vedremo più avanti nel capitolo, il 68020 possiede due versioni differenti dello SSP.
Il processore riempie lo stack dagli indirizzi alti di memoria verso quelli bassi usando le modalità di indirizzamento predecremento e postincremento rispettivamente per le operazioni di push e di pop. Ad esempio, al momento di una chiamata a una subroutine, il processore decrementa lo stack pointer, inserisce sullo stack la word di ordine inferiore del program counter, decrementa ancora lo stack pointer e finalmente inserisce sullo stack la word di ordine più elevato del program counter.
Al momento del ritorno da una subroutine, il processore preleva dallo stack

la word di ordine più elevato del program counter, incrementa lo stack pointer e preleva dallo stack la word di ordine più basso del program counter e finalmente incrementa lo stack pointer. Questa operazione è illustrata nella figura seguente



Lo stack deve essere in grado di gestire le chiamate alle subroutine e alle exception, le quali, ambedue, effettuano push e pop del program counter i cui valori sono di tipo long word. Per questo motivo lo stack deve mantenere un allineamento pari rispetto alle word: ciò significa che ogni parola di stack deve cominciare 32 bit dopo quella precedente. Il processore usa il byte più elevato della parola memorizzata sullo stack quando inserisce o preleva dei byte; il byte di ordine più basso non viene usato. Ciò permette di mantenere i requisiti di indirizzabilità pari dello stack.

PROGRAM COUNTER

Il program counter è un registro special purpose a 32 bit che punta alla prima word dell'istruzione successiva (durante l'esecuzione di un'istruzione), oppure alla parola successiva di un'istruzione (durante la fase di fetch). Dal momento che le istruzioni devono essere allineate su indirizzi pari, il program counter deve sempre contenere un indirizzo pari.

CODICI DI STATO

In modalità utente il processore può accedere al byte di ordine inferiore del registro di stato; questo byte, il *condition code register* (registro del codice di condizione) o CCR, contiene dei bit di flag i cui valori dipendono dai risul-

tati di alcune operazioni quali ad esempio somme, sottrazioni, shift, eccetera.
Il bit di Carry (C) viene settato quando, a seguito di una somma, è necessario effettuare il riporto del bit più significativo oppure quando, a seguito di un'operazione di sottrazione, è necessario chiedere in prestito un bit.
Il bit di Overflow (V) è generato dall'OR esclusivo fra il bit più significativo e i successivi bit di ordine più elevato dell'operando a seguito di operazioni aritmetiche. Il settaggio del bit di Overflow implica che il risultato dell'operazione non può essere rappresentato nella dimensione dell'operando specificata.
Il bit di Zero (Z) viene settato quando il risultato di un'operazione è zero: in tutti gli altri casi esso viene posto a zero.
Il bit Negative (N) viene settato a seconda del valore del bit più significativo del risultato di un'operazione con segno. Quando questo bit è settato, il risultato è negativo: quando invece è nullo, il risultato è positivo o nullo.
Il bit di Extend (X) viene usato nelle operazioni aritmetiche in precisione multipla. Quando viene modificato da un'istruzione, esso è identico al bit di Carry.
I registri della modalità utente vengono illustrati nella Figura 2.1.

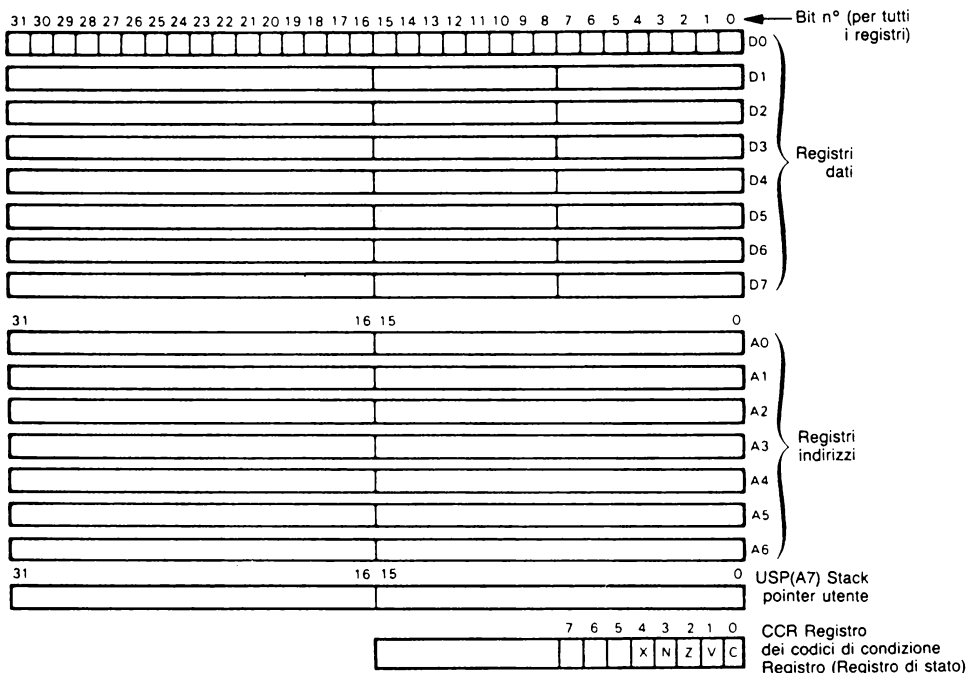


Figura 2.1 Set dei registri in modalità utente.

2.3 Registri in modalità di sistema

Oltre ai registri dati, indirizzi e codici di stato disponibili in modalità utente, i membri della famiglia 68000 hanno parecchi registri addizionali disponibili in modalità supervisore. Dal momento che il tipo e il numero di registri supervisore variano a seconda del particolare membro della famiglia 68000, tratteremo ogni elemento della famiglia singolarmente; a meno che non venga dichiarato espressamente le caratteristiche dei prodotti nella fascia bassa della famiglia 68000 vengono mantenute anche nei prodotti di fascia alta.

68000 e 68008

I due prodotti di fascia bassa, il 68000 e il 68008, posseggono in modalità supervisore due ulteriori registri come mostrato in Figura 2.2. Essi sono il *supervisor stack pointer* (puntatore allo stack in modalità supervisore) che viene identificato con A7' o SSP, e il byte più significativo del registro di stato. Come lo stack pointer utente, il SSP è a 32 bit; la memorizzazione viene effettuata a partire dagli indirizzi alti di memoria verso quelli più bassi: i riferimenti allo stack devono essere effettuati a indirizzi pari. Il processore usa automaticamente il SSP quando si trova in modalità supervisore: il supervisore non può accedere allo stack utente.
Il byte di sistema del registro di stato si combina con il registro dei codici di condizione per formare un registro di stato a 16 bit: il byte di sistema contiene due bit che fungono da flag e tre bit che fungono da maschere di interrupt.
Il bit Supervisor (S) specifica la modalità di esecuzione del processore; quando il processore è settato sta operando in modalità supervisore, se il processore non è settato sta operando in modalità utente.
Il bit di Trace (T) specifica, quando viene settato, che il processore opera in modalità trace. In questa modalità, dopo l'esecuzione di un'istruzione, il processore esegue il processo di exception il cui indirizzo è contenuto nel vetto-

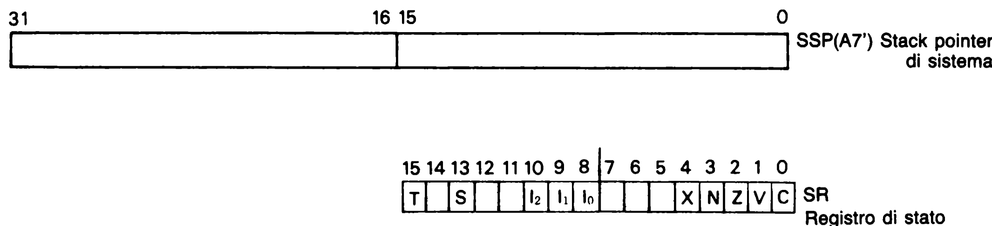


Figura 2.2 Registri addizionali disponibili in modalità supervisore (68000, 68008).

re numero 9 nella tabella dei vettori. Tratteremo la gestione delle exception nel Capitolo 7; per il momento è sufficiente dire che il modo trace implementa una modalità di esecuzione a passo singolo. Ciò permette ad un programma di debugging di controllare i risultati di un programma applicativo istruzione per istruzione.

I processori 68000 possono operare in ognuno degli otto livelli di priorità. I dispositivi esterni possono tentare di interrompere il processore attivando i segnali in ogni combinazione delle linee di richiesta dell'interrupt $\overline{IPL0}$-$\overline{IPL2}$. Il valore binario delle linee di interrupt rappresenta l'importanza del dispositivo che ha generato l'interrupt: un dispositivo ad alta priorità (quale ad esempio un interrupt di clock) possiede un valore più alto rispetto a quello di un dispositivo a bassa priorità, quale ad esempio il driver di un terminale. La priorià più elevata è 7 (111 binario) e la più bassa è 0 (000 binario).

La maschera di interrupt nel registro di stato definisce il livello in cui opera attualmente il processore: nella normale modalità utente esso opera di solito a livello 0. I driver dei vari dispositivi operano di solito a livelli più alti: quando un dispositivo interrompe il processore, la logica interna confronta il valore nella maschera di interrupt con il valore che il dispositivo ha mandato sulle linee $\overline{IPL0}$-$\overline{IPL2}$.

Se le linee di interrupt indicano un livello di priorità più elevato, il processore inizia un processo di exception; se le linee indicano un livello di priorità minore o uguale, il processore ignora temporaneamente la richiesta di interrupt fino a che il codice che è in esecuzione non abbassa la propria priorità (il valore nella maschera di interrupt).

Questa maschera di input, in congiunzione con le altre linee di interrupt, fornisce un ottimo mezzo per implementare i meccanismi di priorità: ciò assicura che le periferiche lente possono interrompere la normale esecuzione di un programma e che dispositivi dipendenti dal tempo possono interrompere periferiche lente.

I processi di exception non si esauriscono in quanto descritto fino a questo punto; il Capitolo 7 descriverà in dettaglio le azioni che il processore e le periferiche debbono intraprendere durante il processo di exception.

68010 e 68012

Il 68010 e il 68012, oltre allo stack pointer a livello di supervisore e il byte di sistema del registro di stato includono un "registro base dei vettori" e due registri per i codici di funzione alternativi. Questi registri addizionali sono illustrati in Figura 2.3.

Il 68000 e il 68008 hanno una tabella di vettori di exception fissa: quest'ultima parte dall'indirizzo 0000H ed è caricata dal software di sistema con gli indirizzi delle routine di gestione delle exception. Il processore sfrutta auto-



Figura 2.3 Registri addizionali disponibili in modalità supervisore (68010, 68012).

maticamente questi indirizzi quando incontra una condizione di exception. Tratteremo in dettaglio queste operazioni nel Capitolo 7; per ora è sufficiente dire che, mentre il 68000 e il 68008 richiedono a livello hardware che la tabella dei vettori inizi all'indirizzo 0000H, il 68010 permette al software di specificare l'indirizzo di partenza della tabella dei vettori mediante il *vector base register* (registro base dei vettori), o VBR.

Come menzionato in precedenza, la famiglia di processori 68000 comprende tre linee di output dei codice di funzione sul chip. Questi codici di funzione indicano ad una periferica qualsiasi il tipo di accesso correntemente attivo sul processore. Di default, queste modalità possono indicare accesso a programmi utente, accesso a dati utente, accesso a programmi supervisore, accesso a dati supervisore e ricezione di interrupt.

I registri dei codici funzione alternativi (SFC e DFC) del 68010 permettono a un programma a livello di sistema di specificare in output il proprio codice funzione durante la fase fetch/memorizzazione della sorgente e della destinazione dell'istruzione MOVES (muovi da/verso registri di stato).

68020

Il 68020 usa tutti i registri dello stato supervisore trattati per gli altri processori 68000. Oltre a questi, il 68020 ha due registri per la gestione della memoria cache che, nel 68020, è integrata su singolo chip, e due ulteriori bit nel registro di stato in modalità supervisore. Tutti i registri addizionali del 68020 sono mostrati nella Figura 2.4.

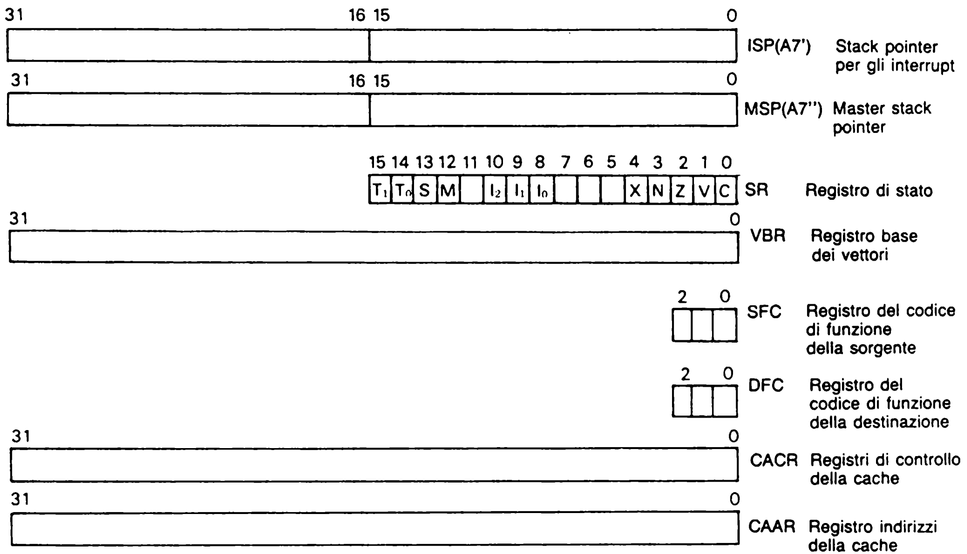


Figura 2.4 Registri addizionali disponibili in modalità supervisore (68020).

I registri di gestione della cache permettono la manipolazione via software della cache istruzioni che, come già detto, è integrata su singolo chip. La cache permette di accedere molto velocemente alle istruzioni, aumentando notevolmente la velocità di esecuzione dei loop. Il *cache control register* (registro di controllo della cache), o CACR, provvede al controllo e all'accesso alla cache istruzioni, mentre il *cache address register* (registro indirizzi della cache), o CAAR, contiene l'indirizzo delle funzioni di controllo della cache.
Il registro di stato del 68020 definisce un secondo bit di abilitazione di trace (T0) che è combinabile con il bit analogo presente negli altri membri della famiglia 68000. I due bit si combinano insieme per fornire indicazioni più precise sul tipo di tracing da effettuare: quando valgono 00 non avviene alcun tracing; quando valgono 01, il tracing avviene solo quando vi è una variazione nel flusso del programma (quale ad esempio l'esecuzione di un branch (diramazione) o di chiamate a subroutine); quando valgono a 10, il processore va in trap dopo ogni istruzione: questa modalità è quella standard negli altri membri della famiglia 68000. Il valore 11 è indefinito.
Il 68020 delega i compiti relativi alla gestione dello stack in modalità supervisore a due stack pointer distinti: l'*interrupt stack pointer* (puntatore allo stack per gli interrupt), o ISP, e *master stack pointer* (puntatore allo stack principale), o MSP. Il bit master (M) contenuto nel registro di stato determina quale puntatore allo stack deve essere utilizzato quando il processore si trova in modalità supervisore (quando è settato il bit S). Quando il bit M è nullo, il processore usa lo ISP: quest'ultima modalità è equivalente a quella degli altri processori 68000. Se il bit M è settato il processore userà il MSP.

Avere due stack pointer differenti in modalità supervisore è molto utile in sistemi multitasking in quanto si permette al sistema operativo di distinguere fra exception asincrone (ad esempio I/O) e chiamate al sistema operativo da parte dell'utente. Ciò permette di implementare un'interfaccia più "pulita", dal momento che il task di un utente potrebbe richiedere di usare routine normalmente riservate al sistema ad esempio attraverso un'istruzione di TRAP. Il master stack può contenere informazioni dipendenti dal particolare task e delle zone di memoria temporanea che servono all'utente. L'ISP può quindi essere usato per trattare esclusivamente exception asincrone che, generalmente, sono indipendenti dal processo in esecuzione e non richiedono quindi informazioni particolari circa il task in esecuzione.

2.4 Organizzazione della memoria

ORDINAMENTO DEI BYTE

La famiglia 68000 permette l'accesso in memoria a livello di byte, word e long word. Un indirizzo specifica la locazione del byte o, per dati che occupano più di un byte, la locazione del byte più significativo. Il byte meno significativo di una word si troverà quindi all'indirizzo specificato più uno; il byte meno significativo di una long word si troverà allora all'indirizzo specificato più 3. La Figura 2.5 mostra l'allineamento in memoria di byte, word e long word.

ALLINEAMENTO

Nei processori 68000, 68008, 68010 e 68012 le word e le long word (parole di istruzioni comprese) devono iniziare a indirizzi pari. Questa è una richiesta dettata dalla particolare conformazione del bus dati. Il tentativo di accedere a una word o a una long word a un indirizzo dispari, darà inizio al processo di exception che descriveremo in dettaglio nel Capitolo 7.
Il 68020 aggira questa limitazione controllando preventivamente gli accessi a word dispari: se tali accessi si verificano esso compie due o più accessi parziali alla word o alla long word. Naturalmente ciò provoca un ulteriore sovraccarico all'esecuzione: il programmatore dovrebbe quindi cercare di allineare le word a indirizzi pari (bit 0 uguale a zero) e long word a indirizzi multipli di quattro (bit 0 e 1 uguali a zero).
Per ottenere la massima efficienza il 68020 mantiene la restrizione tipica della famiglia 68000 circa l'allineamento delle istruzioni a indirizzi pari. Il 68020 usa anche push e pop di long word sul suo stack di sistema: quindi, per operare al meglio, lo stack dovrebbe essere localizzato a un indirizzo pari a un multiplo di quattro.

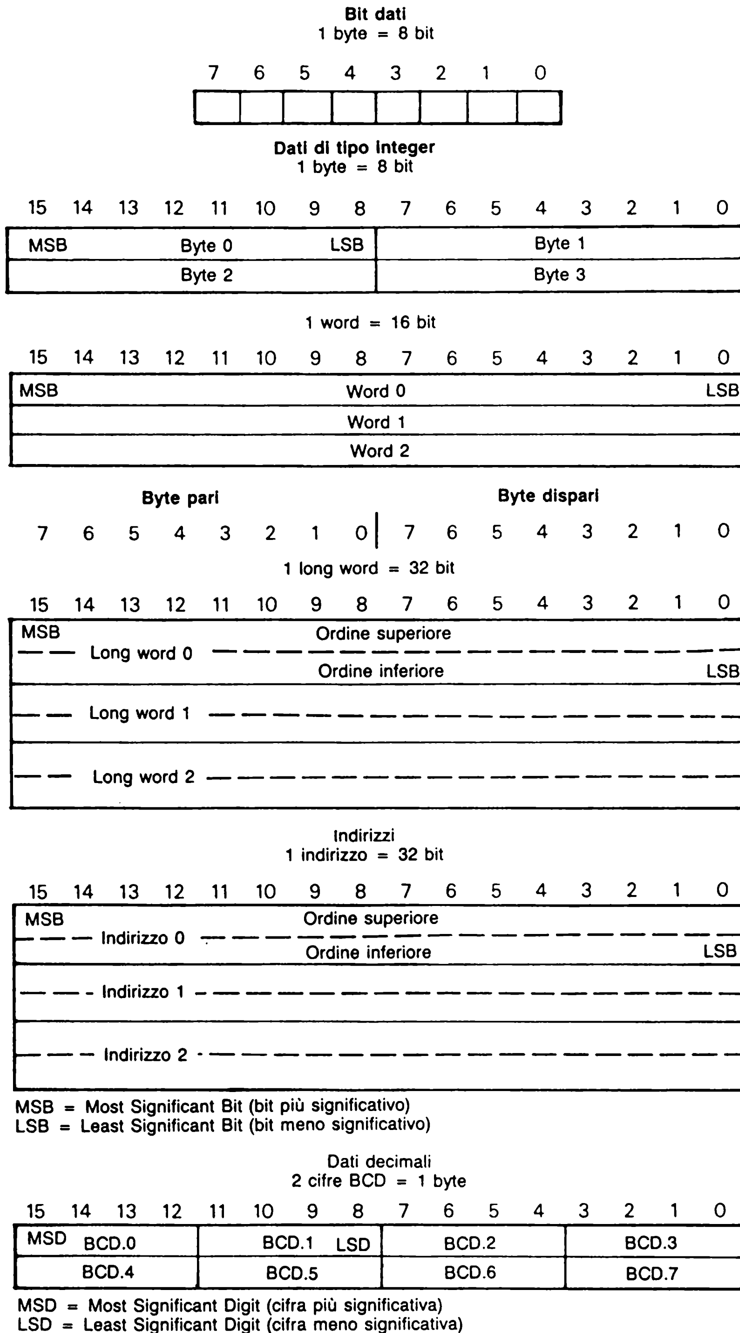


Figura 2.5 Organizzazione dei dati in memoria.

2.5 Memoria virtuale e macchine virtuali

I processori della famiglia 68000 possono accedere a zone molto vaste di memoria. In ogni caso, in molte occasioni, non è molto economico avere tutta la memoria indirizzabile presente fisicamente. Il 68010, 68012 e il 68020 permettono ai programmi di usare tecniche di "memoria virtuale" per simulare la presenza di tutto lo spazio di memoria indirizzabile sulla macchina.

MEMORIA VIRTUALE

Se il processore tenta di accedere a zone di memoria non presenti fisicamente nel sistema, l'unità di gestione della memoria riporta un *page fault* (mancanza di pagina), che comporta l'esecuzione di una exception che si proccupa di gestire l'errore che ne risulta. A questo punto il sistema operativo si interroga sul tipo di errore che è stato commesso: se l'errore è dovuto al fatto che il programma stava tentando di accedere a una zona di memoria che temporaneamente si trova in una memoria secondaria (ad esempio su disco), il sistema operativo può leggere dal disco il blocco di memoria necessario e trasferirlo nella memoria fisica.
Una volta che il dato necessario è in memoria, il sistema restituisce il controllo al programma utente: il processore usa quindi la *continuazione* di una istruzione per completare l'istruzione interrotta. Usando questo metodo il processore continua l'esecuzione del programma partendo dal punto in cui si era riscontrato il page fault (spesso ciò avviene all'interno di un'istruzione). Per trattare correttamente la continuazione, il processore deve salvare alcune informazioni interne prima di eseguire il codice di exception. Il 68010, 68012 e il 68020 salvano queste informazioni sullo stack prima dell'esecuzione della exception e restituiscono queste informazioni al processore dopo aver eseguito l'istruzione di ritorno dall'exception (RTE).

MACCHINE VIRTUALI

Nello sviluppo di sistemi, può essere necessario scrivere del codice che sia in grado di accedere a un dispositivo che non è presente fisicamente nel vostro sistema, ad esempio una porta di comunicazione seriale. Potreste anche avere bisogno di eseguire istruzioni che il vostro processore non permette, ad esempio istruzioni caratteristiche di un coprocessore floating point. Una "macchina virtuale" forza l'esecuzione di exception quando si tenta di accedere a indirizzi di memoria non definiti (ad esempio il registro di stato di un dispositivo mancante) o quando viene comunicato al processore di eseguire un'istruzione che non è presente nel suo set di istruzioni. Questo dispositivo permette, scrivendo un apposito codice di gestione delle exception, di emulare il dispositivo o l'istruzione via software.

3

Modalità di indirizzamento

La famiglia 68000 utilizza 18 tipi diversi di indirizzamento che possono essere raggruppati in 7 categorie:

1. Indirizzamento diretto a registro
   a. A registro dati diretto
   b. A registro indirizzi diretto

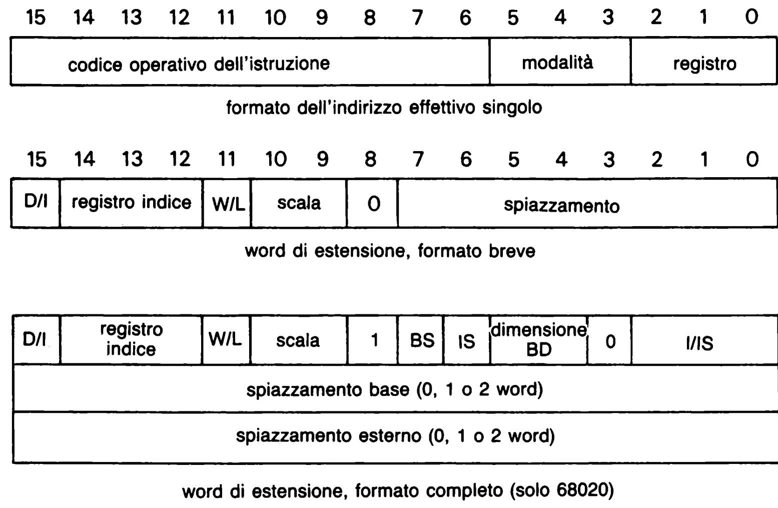
2. Indiretto a registro
   a. Registro indirizzi indiretto
   b. Registro indirizzi indiretto con postincremento
   c. Registro indirizzi indiretto con predecremento
   d. Registro indirizzi indiretto con spiazzamento
   e. Registro indirizzi indiretto con indice e spiazzamento di 8 bit*

3. Indiretto a memoria
   a. Indiretto a memoria postindicizzato*
   b. Indiretto a memoria preindicizzato*

4. Indirizzamento indiretto relativo al program counter (PC)
   a. PC indiretto con spiazzamento di 16 bit
   b. PC indiretto con indice e spiazzamento di 8 bit*
   c. PC indiretto con indice e spiazzamento di 16 0 32 bit*

5. Indirizzamento indiretto in memoria relativo al program counter (PC)
   a. Indirizzamento in memoria indiretto rispetto al PC postindicizzato*
   b. Indirizzamento in memoria indiretto rispetto al PC preindicizzato*

Tabella 3.1 Modalità di indirizzamento.



registro — Registro dati o indirizzi (vedi Tabella 3.2)

modalità — Modalità di indirizzamento (vedi Tabella 3.2)

codice operativo — Informazioni sulle istruzioni e possibili informazioni modalità/registro sul secondo operando

spiazzamento — Valore 8 bit con segno

scala — Fattore di scala indicizzato (solo 68020)
00 = 1X
01 = 2X
10 = 4X
11 = 8X

I/IS — Selezione indice/indiretto (solo 68020, vedi Tabella 3.3)

BD — Dimensione dello spiazzamento base (solo 68020)
00 Riservato
01 Spiazzamento nullo
10 Spiazzamento di una word
11 Spiazzamento di una long word

IS — Soppressione dell'indice (solo 68020, vedi Tabella 3.3)

BS — Soppressione della base (solo 68020)
0 Valuta e aggiungi al registro base
1 Sopprimi il registro base

W/L — Dimensione del registro indice
0 Word estesa con segno
1 Long word con segno

registro indice — Registro dati o indirizzi (000-111)

D/I — Tipo del registro indice
0 Registro dati
1 Registro indirizzi

6. Assoluto
   a. Assoluto corto
   b. Assoluto lungo

7. Immediato

Notate che le modalità segnate con asterisco (*) svolgono funzioni particolari sul 68020 oppure sono peculiari del 68020. Per maggiori dettagli rimandiamo alla trattazione nel corso del capitolo.

3.1 Codifica degli indirizzi

Le istruzioni dei processori della famiglia 68000 codificano gli indirizzi degli operandi nel modo seguente: la prima word specifica l'operazione e, in molti casi, l'indirizzo di un operando; ulteriori specifiche relative agli operandi seguono questa prima parola. Difatti i primi quattro membri della famiglia 68000 permettono estensioni fino a 10 word, la maggior parte delle quali serve per l'implementazione delle varie modalità di indirizzamento. Gli indirizzi vengono codificati usando un campo *modalità* e un campo *registro*: ognuno di questi campi è lungo 3 bit ed è contenuto nella prima word dell'istruzione. Ogni parola che segue può rappresentare semplicemente l'in-

Tabella 3.2 Codifica modalità/registro.

Modalità	Registro	Operazione di indirizzamento
000	reg n.	Registro dati diretto
001	reg n.	Registro indirizzi diretto
010	reg n.	Registro indirizzi indiretto
011	reg n.	Registro indirizzi indiretto con postincremento
100	reg n.	Registro indirizzi indiretto con preincremento
101	reg n.	Registro indirizzi indiretto con spiazzamento
110	reg n.	Registro indirizzi indiretto in memoria con indice*
111	000	Assoluto corto
111	001	Assoluto lungo
111	010	Program counter indiretto con spiazzamento
111	011	Program counter indiretto in memoria con indice*
111	100	Registro dati immediato
111	101-111	Riservato

*Solo 68020

dirizzo dell'operando, nel caso dell'indirizzamento assoluto, o può essere piuttosto complessa e contenere campi di indirizzamento come ad esempio nel caso di modalità di indirizzamento indiretto. I campi più complessi delle estensioni indicano la presenza di indicizzazione e di indirezione oltre alla presenza di un indirizzo e di uno spiazzamento.
La Tabella 3.1 descrive i campi delle varie modalità di indirizzamento all'interno della word delle istruzioni e all'interno delle word delle estensioni. La Tabella 3.2 descrive i valori modalità/registro e le loro definizioni. La Tabella 3.3 descrive le definizioni di index suppress e la scelta indice/indiretto. Il significato dell'accoppiamento modalità/registro e dei bit di indice e di indirezione dovrebbero risultare evidenti nelle modalità di indirizzamento descritte nel seguito.

Tabella 3.3 Codifica della modalità di indirizzamento IS-I/S (solo 68020).

IS	I/IS	Operazioni di indirizzamento
0	000	Indice, senza indirezione in memoria
0	001	Indiretto preindice con nessuno spiazzamento esterno
0	010	Indiretto preindice con spiazzamento esterno di una word
0	011	Indiretto preindice con spiazzamento esterno di una longword
0	100	Riservato
0	101	Indiretto postindice con nessuno spiazzamento esterno
0	110	Indiretto postindice con spiazzamento esterno di una word
0	111	Indiretto postindice con spiazzamento esterno di una longword
1	000	Nessun indice, senza indirezione in memoria
1	001	Nessun indice, indiretto in memoria con nessuno spiazzamento esterno
1	010	Nessun indice, indiretto in memoria con spiazzamento esterno di una word
1	011	Nessun indice, indiretto in memoria con spiazzamento esterno di una longword
1	100-111	Riservato

3.2 Modalità di indirizzamento

REGISTRO DATI DIRETTO

Nella modalità a registro dati diretto, un registro dati contiene l'operando:

Registro dati: operando

Sintassi assembler: Dn

REGISTRO INDIRIZZI DIRETTO

Nella modalità a registro indirizzi diretto, un registro indirizzi contiene l'operando:

Registro indirizzi: operando

Sintassi assembler: An

REGISTRO INDIRIZZI INDIRETTO

Nella modalità a registro indirizzi indiretto, un registro indirizzi contiene l'indirizzo dell'operando; in altre parole esso *punta* all'operando.

Registro indirizzi: indirizzo di memoria

punta a

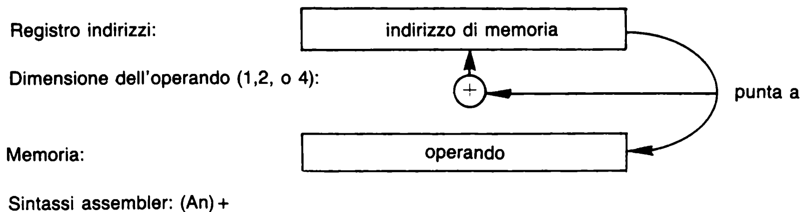
Memoria: operando

Sintassi assembler: An

REGISTRO INDIRIZZI INDIRETTO CON POSTINCREMENTO

Come nella modalità a registro indirizzi indiretto semplice, un registro indirizzi in modalità indiretta con postincremento contiene l'indirizzo dell'operando. In ogni caso, dopo l'uso dell'operando, il processore aggiunge la lunghezza dell'operando al registro indirizzi (1 per operazioni di 1 byte, 2 per operazioni di tipo word o 4 per operazioni di tipo long word).

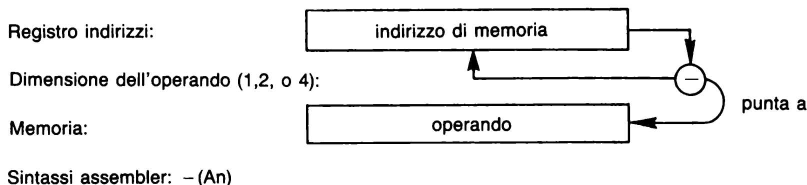
Nota: Se il registro indirizzi è lo stack pointer e l'operazione ha le dimensioni di un byte, il processore aggiunge automaticamente 2 al registro. Questa operazione forza lo stack a essere word-allineato per permettere il raggiungimento del massimo dell'efficienza.



REGISTRO INDIRIZZI INDIRETTO CON PREDECREMENTO

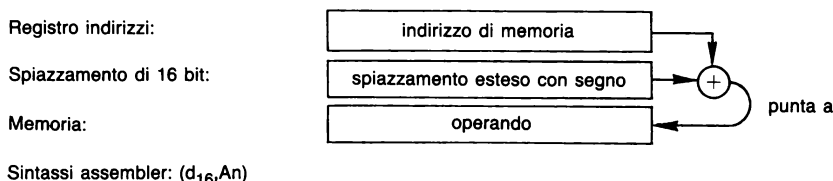
Come nella modalità a registro indirizzi indiretto semplice, un registro indirizzi con predecremento contiene l'indirizzo dell'operando. In ogni caso, prima di usare l'operando, il processore sottrae la lunghezza dell'operando dal registro indirizzi (cioè 1 per operazioni di tipo byte, 2 per operazioni di tipo word o 4 per operazioni di tipo long word).

Nota: Se il registro indirizzi è lo stack pointer e l'operazione ha le dimensioni di un byte, il processore sottrae automaticamente 2 dal registro. Ciò forza lo stack a restare word-allineato per raggiungere il massimo dell'efficienza.



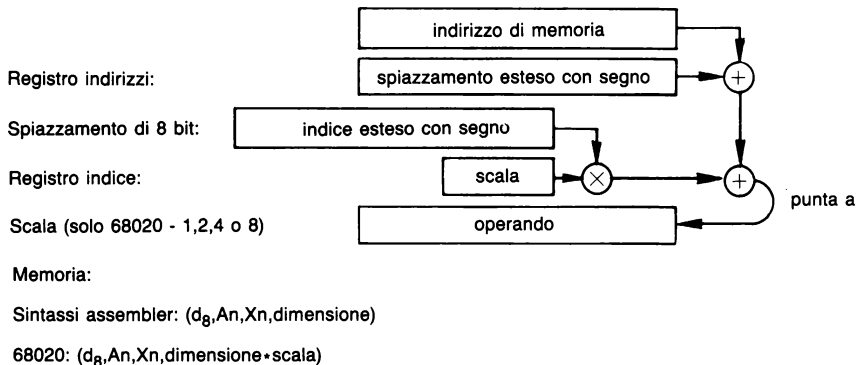
REGISTRO INDIRIZZI INDIRETTO CON SPIAZZAMENTO

Nella modalità a registro indirizzi indiretto con spiazzamento, l'operando risiede all'indirizzo dato dalla somma fra il contenuto di un registro indirizzi e di uno spiazzamento a 16 bit. Il valore dello spiazzamento è esteso con segno a 32 onde per permettere spiazzamenti negativi.



REGISTRO INDIRIZZI INDIRETTO CON INDICE E SPIAZZAMENTO

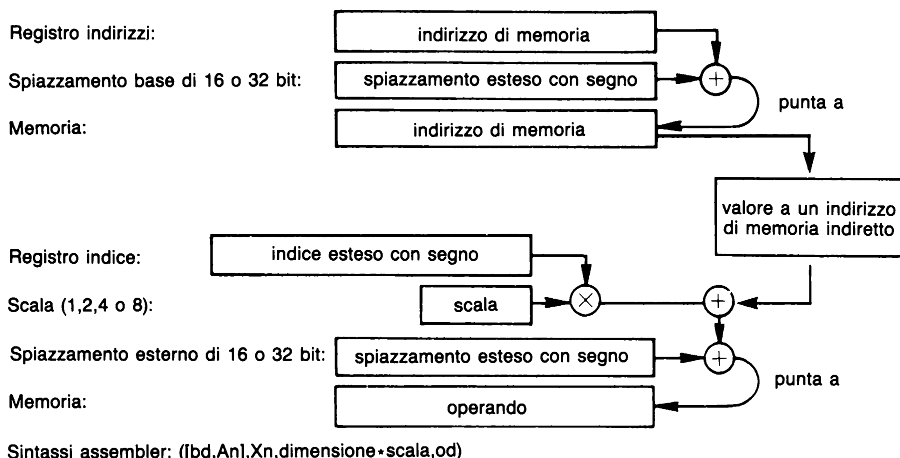
Nella modalità indiretta con indice e spiazzamento, l'operando risiede all'indirizzo dato dalla somma fra il contenuto di un registro indirizzi, uno spiazzamento e il prodotto di un registro indice e un fattore di scala.
Se lo spiazzamento base è di 8 o 16 bit, esso viene esteso con segno a 32 bit. (La versione a 8 bit di questa modalità di indirizzamento si trova solo nel 68020.) Il valore del registro indice può essere un valore a 16 o 32 bit; se è un valore a 16 bit, il processore estende il valore a 32 bit con segno prima di effettuare l'operazione di somma. Il fattore di scala potrebbe essere 2^0 (1), 2^1 (2), 2^2 (4), o 2^3 (8). Solo il 68020 supporta quest'ultimo tipo di indicizzazione.



INDIRETTA MEMORIA POSTINDICIZZATO

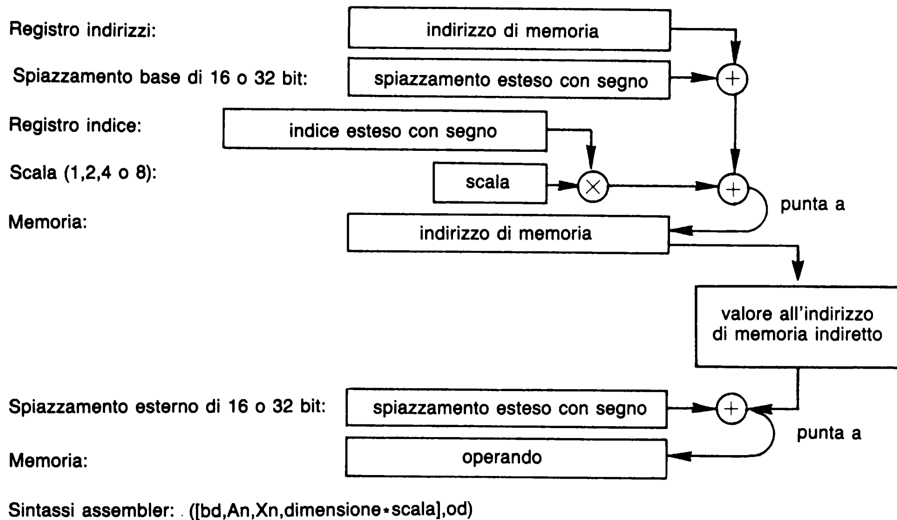
Nella modalità di indirizzamento indiretto a memoria postindicizzato (esclusivo del 68020), l'indirizzo dell'operando comprendeva quattro valori: il contenuto di un registro indirizzo, uno spiazzamento di 16 o 32 bit, il prodotto del valore in un registro indice e la scala, e un secondo (esterno) spiazzamento. La base e lo spiazzamento esterno possono essere di 16 o 32 bit; se sono di 16 bit, il processore estende con segno il valore prima di effettuare la somma. Il registro indice è dimensionato ed esteso con segno (come richiesto) prima di essere sommato all'indirizzo effettivo. Tutti e quattro i valori sono opzionali; se vengono omessi il processore assume valore zero come contributo all'indirizzo effettivo.
Nel valutare l'indirizzo effettivo, il processore aggiunge il valore contenuto nel registro indirizzi allo spiazzamento base; questo valore viene poi usato come indirizzo di un secondo valore. A questo secondo valore il processore

aggiunge il valore aggiornato secondo il valore di scala nel registro indice. Finalmente esso lo aggiunge allo spiazzamento esterno per ottenere l'indirizzo dell'operando.



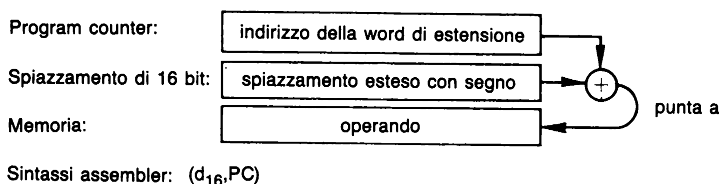
INDIRETTO A MEMORIA PREINDICIZZATO

Nella modalità di indirizzamento indiretto preindicizzato (esclusiva del 68020), l'indirizzo dell'operando comprende quattro valori: il contenuto di un registro indirizzo, uno spiazzamento di 16 o 32 bit, il prodotto fra il valore in un registro indice e la scala e un secondo (esterno) spiazzamento.
La base e lo spiazzamento esterno possono essere di 16 o 32 bit; se sono di 16 bit, il processore estende con segno il valore prima di sommarlo. Il registro indice è dimensionato ed esteso con segno (come richiesto) prima di venire sommato all'indirizzo effettivo. Tutti e quattro i valori sono opzionali: se vengono omessi il processore considera il relativo valore pari a 0.
Nella valutazione dell'indirizzo effettivo, il processore somma il valore del registro indirizzo allo spiazzamento base e il valore aggiornato secondo il valore di scala dal registro indice. Esso usa questo valore come l'indirizzo di un secondo valore: a questo secondo valore il processore aggiungerà poi lo spiazzamento esterno per ottenere l'indirizzo dell'operando.



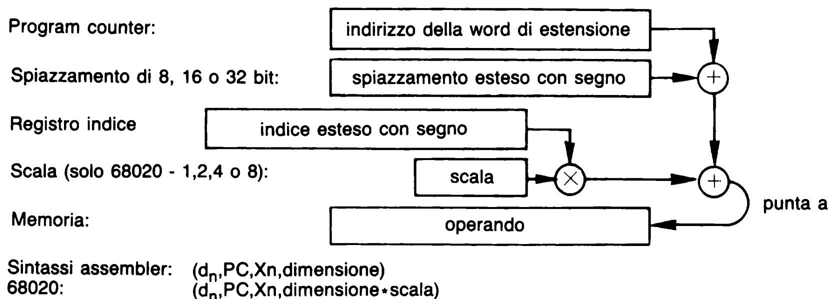
PC INDIRETTO CON SPIAZZAMENTO

Nella modalità indiretta con spiazzamento, il processore genera un indirizzo effettivo sommando il valore del program counter al valore esteso con segno della word di estensione a 16 bit.



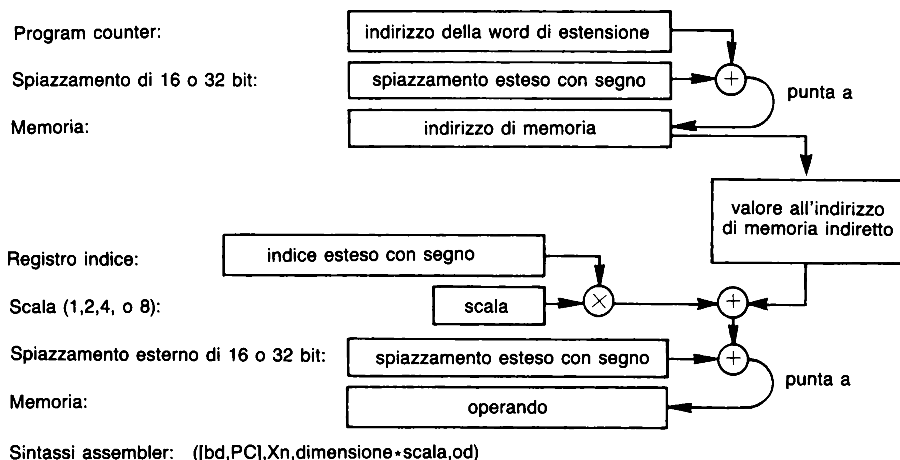
PC INDIRETTO E SPIAZZAMENTO

Nell'indirizzamento a PC indiretto con indice e spiazzamento, l'indirizzo effettivo è dato dalla somma fra l'indirizzo contenuto nel program counter, il valore della word di estensione e il valore del registro indice (moltiplicato per il fattore di scala nel caso del 68020). Il valore nel registro indice può essere esteso con segno, se necessario, e (nel caso del 68020) moltiplicato per il fattore di scala.



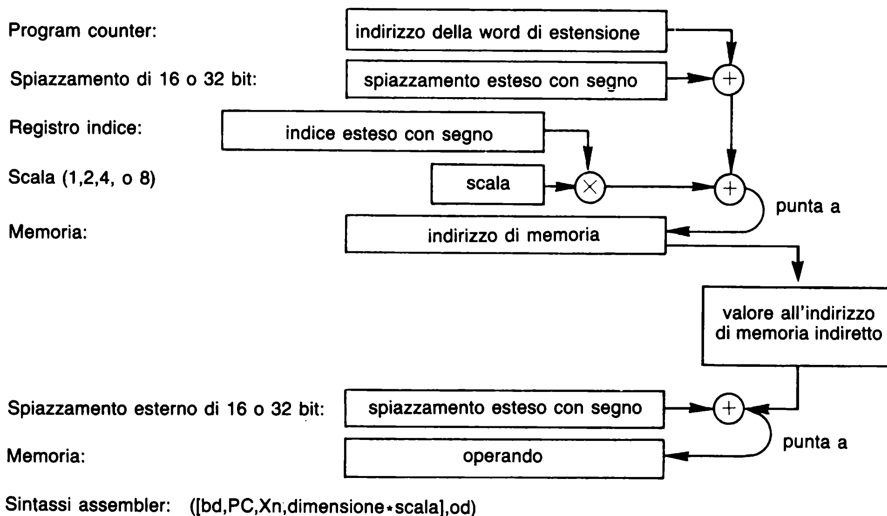
INDIRIZZAMENTO IN MEMORIA INDIRETTO RISPETTO AL PC POSTINDICIZZATO

Nell'indirizzamento in memoria indiretto rispetto al PC con postindicizzato, il calcolo dell'indirizzo effettivo richiede 4 valori; la somma fra il PC e lo spiazzamento base viene usata come indirizzo. Il processore somma il valore contenuto in quell'indirizzo con il valore contenuto in un registro indice opportunamente aggiornato secondo un fattore di scala e lo spiazzamento esterno per ottenere l'indirizzo dell'operando.
Il valore del program counter è quello pari all'indirizzo della word di estensione: la base e gli spiazzamenti più esterni possono essere di 16 o 32 bit; se sono di 16 bit, il processore estende con segno il valore prima di usarlo. Il valore del registro indice può avere la dimensione di byte, word o long word e può essere esteso con segno in caso di necessità. Il valore di scala può essere 1, 2, 4 o 8. Tutte le componenti usate per il calcolo dell'indirizzo dell'operando sono opzionali; se vengono omessi, il processore assume 0 come valore di default.



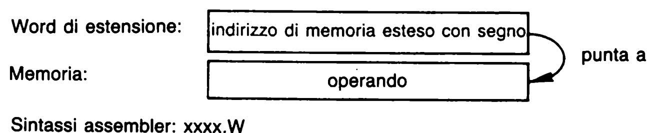
INDIRIZZAMENTO IN MEMORIA INDIRETTO RISPETTO AL PC PREINDICIZZATO

Nell'indirizzamento in memoria indiretto rispetto al PC preindicizzato, il calcolo dell'indirizzo effettivo richiede 4 valori; la somma fra il PC e lo spiazzamento base viene usata come indirizzo. Il processore somma il valore contenuto in quell'indirizzo con lo spiazzamento esterno per ottenere l'indirizzo dell'operando.
Il valore del program counter è quello dell'indirizzo della word di estensione: la base e gli spiazzamenti più esterni possono essere di 16 o 32 bit; se sono di 16 bit, il processore estende con segno il valore prima di usarlo. Il valore del registro indice può avere la dimensione di byte, word o long word e può essere esteso con segno in caso di necessità. Il valore di scala può essere 1, 2, 4 o 8. Tutte le componenti usate per il calcolo dell'indirizzo dell'operando sono opzionali; se vengono omessi il processore assume 0 come valore di default.



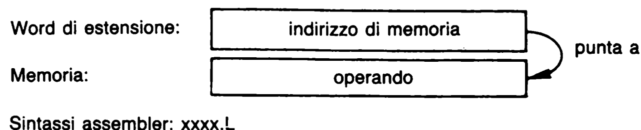
ASSOLUTO CORTO

Con l'indirizzamento assoluto corto, l'indirizzo dell'operando è il valore esteso con segno della word di estensione di 16 bit che segue l'istruzione.



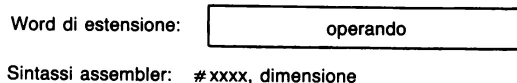
ASSOLUTO LUNGO

Nell'indirizzamento assoluto lungo, l'indirizzo dell'operando è rappresentato dalla long word che segue la word relativa all'istruzione.



INDIRIZZAMENTO IMMEDIATO

Nell'indirizzamento immediato, l'operando stesso segue la word relativa all'istruzione. A seconda delle dimensioni dell'operando, l'istruzione richiede una o due word di estensione.



4

Il set di istruzioni

Il 68000 ha un set di più di 300 istruzioni, molte delle quali sono simili fra loro e differiscono solo nel tipo di dati che trattano o nella modalità di indirizzamento che usano. In realtà, le istruzioni base del 68000 sono solo una cinquantina e variano di poco tra i vari membri della famiglia. Il set di istruzioni dovrebbe essere relativamente semplice da imparare dal momento che le differenze fra le varie famiglie sono piuttosto consistenti e non vi è quindi pericolo di confusione.
Il formato base di tutte le istruzioni è lo stesso sia internamente sia dal punto di vista del codice mnemonico ad esse assegnato. Internamente, il codice operativo di ogni istruzione è di una word e i vari modi di indirizzamento sono racchiusi in esso. Certe modalità di indirizzamento richiedono estensioni alla word dell'istruzione per specificare l'operando (attraverso spiazzamenti, indicizzazione o indirezione). A seconda della modalità, l'istruzione totale (codice operativo più word di estensione) può essere lunga da uno fino a undici word.

COMPATIBILITÀ VERSO L'ALTO

Motorola ha scelto di rendere ogni membro della famiglia 68000 compatibile verso l'alto. Ciò significa che il codice del 68000 girerà anche sul 68020; il viceversa però non vale, nel senso che il codice scritto per il 68020 non necessariamente girerà sul 68000. Questa filosofia è piuttosto valida, in quanto la migrazione verso l'alto è spesso dettata dal desiderio di migliorare la velocità e la potenza, mentre la migrazione verso il basso è dettata dal desiderio di diminuire il costo dell'applicazione. Un progetto a basso costo co-

struito per un 68008 potrebbe non richiedere l'uso della memoria cache, delle istruzioni del coprocessore o delle modalità di indirizzamento più avanzate supportate dal 68020.

4.1 Tipi di istruzioni

Le varie istruzioni possono essere raggruppate in 10 categorie base:

- Movimento dati
- Aritmetica intera
- Aritmetica booleana
- Shift e Rotate
- Manipolazione di bit individuali
- Manipolazione di campi di bit
- Manipolazione di stringhe BCD (Binary Coded Decimal)
- Controllo del flusso di istruzioni
- Controllo del sistema
- Comunicazioni multitask/multiprocessore

Notate che non vi sono istruzioni dedicate all'I/O. A differenza di altri microprocessori, la famiglia 68000 usa un tipo di I/O mappato in memoria invece che tramite porta. Ciò significa che il processore comunica con i dispositivi di I/O come se comunicasse con una qualsiasi area di memoria, rendendo quindi più facile il compito sia del programmatore sia del progettista del sistema.

MOVIMENTO DATI

Le istruzioni di movimento dei dati provvedono allo spostamento di dati fra registro e registro, fra registro e locazione di memoria e direttamente fra due locazioni di memoria. Le istruzioni permettono la manipolazione di dati, indirizzi, scambi fra registri dati, caricamento e memorizzazione da registri multipli, linking e unlinking di stack frame.
Vi sono due cose molto importanti da notare: in primo luogo che non vi sono operazioni di PUSH e POP. Nonostante ciò è possibile effettuare operazioni di stack tramite una istruzione di MOVE usando l'indirizzamento con autoincremento e autodecremento con (A7)+ e −(A7).
In secondo luogo, a differenza di qualche microprocessore, non vi sono istruzioni per lo spostamento di blocchi. È possibile accelerare lo spostamento di blocchi usando la modalità di autoincremento e autodecremento unitamente a un'istruzione di decrement and branch (decremento e ramificazione) il cui mnemonico è DBcc.

Tabella 4.1 Istruzioni di movimento dati.

Mnemonico	Operazione
EXG	(Exchange) Scambia i registri
LEA	(Load Effective Address) Carica l'indirizzo effettivo
LINK	Linka e alloca lo stack
MOVE	Sposta la sorgente nella destinazione
MOVEA	Sposta la sorgente in un registro indirizzi
MOVEC	Sposta i registri di controllo
MOVEM	Sposta più registri
MOVEP	Sposta verso una periferica
MOVEQ	Sposta dati brevi verso la destinazione
MOVES	Sposta lo spazio indirizzi
PEA	(Push Effective Address) Push sullo stack dell'indirizzo effettivo
UNLK	(UnLink) Rimuove il link allo stack

Il 68010 e 68012 incrementano la velocità di trasferimento dei blocchi usando istruzioni ciclabili; il 68020 accelera queste operazioni grazie all'uso della sua cache interna. Entrambi questi concetti sono trasparenti al programmatore e verranno trattati verso la fine del capitolo.
La Tabella 4.1 riassume le istruzioni di movimento dei dati.

ARITMETICA INTERA

Il 68000 fornisce le 4 funzioni aritmetiche di base ovvero addizione, sottrazione, moltiplicazione e divisione, unitamente a operazioni aritmetiche per il confronto di interi, per azzerare un intero, per negare un intero e per effettuare operazioni in precisione multipla. È possibile sommare, confrontare e sottrarre sia indirizzi che dati; la manipolazione degli indirizzi è ristretta a valori di 16 e 32 bit, mentre la manipolazione dei dati include valori di 8 bit.
Le operazioni di moltiplicazione e di divisione possono essere eseguite con o senza segno a seconda di come è necessario. Tutti i membri della famiglia 68000 possono moltiplicare due interi di 16 bit fornendo un risultato di 32 bit e possono dividere un dividendo di 32 bit per un divisore di 16 bit fornendo un quoziente e un resto di 16 bit. Il 68020 permette di moltiplicare elementi di 32 bit ottenendo un risultato di 64 bit e di dividere dividendi di 64 bit per divisori di 32 bit ottenendo un quoziente e un resto di 32 bit.
La Tabella 4.2 riassume le istruzioni di aritmetica intera.

Tabella 4.2 Istruzioni di aritmetica intera.

Mnemonico	Operazione
ADD	Somma la sorgente alla destinazione
ADDA	Somma la sorgente al registro indirizzi
ADDI	Somma un dato immediato alla destinazione
ADDQ	Somma un dato corto alla destinazione
ADDX	Somma con il bit di estensione alla destinazione
CLR	(Clear) Cancella l'operando
CMP	(Compare) Confronta la sorgente con la destinazione
CMPA	(Compare Address) Confronta la sorgente con il registro indirizzi
CMPI	Confronta dati immediati con la destinazione
CMPM	(Compare Memory) Confronta la memoria
CMP2*	Confronta un registro ai limiti inferiore/superiore
DIVS	Divisione con segno
DIVU	Divisione senza segno
DIVSL*	Divisione fra long con segno
DIVUL*	Divisione fra long senza segno
EXT	(Extend) Estensione con segno
EXTB	(Extend Byte) Byte esteso con segno
MULS	(Signed Multiplication) Moltiplicazione con segno
MULU	(Unsigned Multiplication) Moltiplicazione senza segno
NEG	Negazione
NEGX	Negazione con estensione
SUB	Sottrai la sorgente dalla destinazione
SUBA	Sottrai la sorgente dal registro indirizzi
SUBI	Sottrai un dato immediato dalla destinazione
SUBQ	Sottrai un dato corto dalla destinazione
SUBX	Sottrai con il bit di estensione dalla destinazione
*solo 68020	

ARITMETICA BOOLEANA

I processori della famiglia 68000 permettono le operazioni logiche di AND, OR, XOR (OR esclusivo) e NOT: di queste istruzioni fa parte anche un'istruzione di test che confronta un intero qualsiasi con lo zero e setta un apposito flag del registro del codice di condizione in conformità con il risultato ottenuto. L'istruzione Scc testa questo registro e setta una locazione di memoria a uno o a zero a seconda dello stato dei codici.
La Tabella 4.3 riassume le istruzioni booleane.

Tabella 4.3 Istruzioni booleane.

Mnemonico	Operazione
AND	AND fra sorgente e destinazione
ANDI	AND fra dato immediato e destinazione
EOR	OR esclusivo fra sorgente e destinazione
EORI	OR esclusivo fra dato immediato e destinazione
NOT	NOT con destinazione
OR	OR fra sorgente e destinazione
ORI	OR fra un dato immediato e destinazione
Scc	Test dei codici di condizione e settaggio dell'operando
TST	Test dell'operando e settaggio dei codici di condizione

SHIFT E ROTATE

I processori della famiglia 68000 permettono shift aritmetici e logici (sia a destra che a sinistra) e rotazioni con o senza estensione di segno (anche in questo caso le rotazioni possono avvenire sia a sinistra che a destra).
Gli shift aritmetici differiscono dagli shift logici per quanto riguarda il codice di condizione settato dall'operazione; inoltre gli shift aritmetici mantengono il contenuto del bit più significativo (quello del segno), mentre gli shift logici a destra forzano uno zero nel bit più significativo.
Una rotazione senza estensione ha l'effetto di spostare i bit che compongono il numero intero a sinistra o a destra, con i bit che compongono il numero che, per effetto della rotazione, escono dall'intero dalla parte opposta della rotazione. La rotazione con estensione differisce dalla precedente nel senso che i bit che escono dalla word vanno a finire nel flag di estensione del regi-

Tabella 4.4 Istruzioni di shift e rotazione.

Mnemonico	Operazione
ASL	(Arithmetic Shift Left) Shift aritmetico a sinistra
ASR	(Arithmetic Shift Right) Shift aritmetico a destra
LSL	(Logical Shift Left) Shift logico a sinistra
LSR	(Logical Shift Right) Shift logico a destra
ROL	(Rotate Left) Rotazione a sinistra
ROR	(Rotate Right) Rotazione a destra
ROXL	(Rotate Extend Left) Rotazione a sinistra con bit di estensione
ROXR	(Rotate Extend Right) Rotazione a destra con bit di estensione
SWAP	Scambia le word di una longword

stro del codice di condizione e il bit di estensione rientra dal lato opposto della word.
La Tabella 4.4 riassume le istruzioni di shift e rotazione.

MANIPOLAZIONE DI BIT INDIVIDUALI

I processori della famiglia 68000 permettono a un programma di testare, azzerare, settare e invertire (negazione logica) un bit individuale di un valore intero. Le operazioni di modifica, pulizia e settaggio sono utili in sistemi multitasking in quanto combinano la lettura e la scrittura nella stessa istruzione rendendo possibile ad un programma il controllo dell'accesso a una qualsiasi struttura dati.
Queste istruzioni non sono però "indivisibili"; un programma quindi non dovrebbe dipendere da esse per escludere altri processori in un sistema a multiprocessore, dal momento che il processore che attualmente è in esecuzione potrebbe cedere il bus a un secondo processore durante l'esecuzione dell'istruzione. Un elenco delle istruzioni indivisibili verrà indicato più avanti sotto la voce "Comunicazioni multitask/multiprocessore".
La Tabella 4.5 riassume le istruzioni di manipolazione di bit.

Tabella 4.5 Istruzioni di manipolazione di bit.

Mnemonico	Operazione
BCHG	(Bit Change) Modifica un bit
BCLR	(Bit Clear) Pulisce un bit
BSET	(Bit Set) Setta un bit
BTST	(Bit Test) Testa un bit

MANIPOLAZIONE DI CAMBI DI BIT

Il 68020 permette ad un programma di modificare "stringhe" o "campi" di bit consecutivi proprio come se fossero bit individuali: le istruzioni possono operare su campi composti anche da 32 bit. Un programma è quindi in grado di inserire ed estrarre, settare, modificare, aggiungere, sottrarre bit e di ricercare sottostringhe di bit all'interno di un campo.
La Tabella 4.6 riassume le istruzioni di manipolazione di campi di bit.

Tabella 4.6 Istruzioni sui campi di bit.

Mnemonico	Operazione
BFCHG*	(Bit Field Change) Modifica di un campo di bit
BFCLR*	(Bit Field Clear) Pulisce un campo di bit
BFEXTS*	(Bit Field Extract and Sign) Estrae ed estende con segno un campo di bit
BFEXTU*	(Bit Field Extract Unsigned) Estrae ed estende con zero un campo di bit
BFFFO*	(Bit First Find set in Field) Trova il primo set di bit in un campo di bit
BFINS*	(Bit Field Insert) Inserimento di un campo di bit
BFSET*	(Bit Field Set) Settaggio di un campo di bit
BFTST*	(Bit Field Test) Test di un campo di bit
*solo 68020	

MANIPOLAZIONE DI STRINGHE BCD

I processori della famiglia 68000 permettono di aggiungere o sottrarre stringhe BCD: il 68020 include istruzioni per convertire stringhe BCD packed in stringhe BCD non packed.
La Tabella 4.7 riassume le istruzioni di manipolazione BCD.

Tabella 4.7 Istruzioni BCD (Binary Coded Decimal).

Mnemonico	Operazione
ABCD	Somma la sorgente alla destinazione
NBCD	Nega la destinazione
PACK*	Comprime la sorgente nella destinazione
SBCD	Sottrae la sorgente dalla destinazione
UNPK*	Decomprime la sorgente nella destinazione
*solo 68020	

CONTROLLO DEL FLUSSO DI PROGRAMMA

Le istruzioni di controllo del flusso permettono a un programma di saltare in modo condizionato o incondizionato verso un altro punto del codice. Le istruzioni permettono di effettuare questi salti sia rispetto al PC sia rispetto a indirizzi assoluti. Tutti i membri della famiglia 68000 permettono la chiamata di subroutine e il successivo rientro attraverso lo stack e, inoltre, il

Tabella 4.8 Istruzioni di controllo del flusso.

Mnemonico	Operazione
Bcc	(Branch conditionally) Diramazione condizionata
BRA	(Branch) Diramazione incondizionata
BSR	(Branch Subroutine) Diramazione verso una subroutine
CALLM*	(Call Module) Chiamata di un modulo
DBcc	(Decrement and Branch conditionally) Test, decremento e diramazione
JMP	(Jump) Salto a un determinato indirizzo
JSR	(Jump to Subroutine) Salta a una subroutine
NOP	(No Operation) Nessuna operazione
RTD**	(Return and Deallocate) Ritorna e dealloca
RTE+	Ritorno da exception
RTM*	Ritorno da modulo
RTR	Ritorna e ripristina i codici di condizione
RTS	Ritorna da subroutine
+ istruzione privilegiata *solo 68020 **solo 68010-68020	

68020 permette l'uso di stack frame formali per facilitare il passaggio dei parametri alle subroutine.
La Tabella 4.8 riassume le istruzioni di controllo del flusso di programma.

CONTROLLO DEL SISTEMA

Le istruzioni di controllo del sistema comprendono istruzioni privilegiate, ovvero quelle istruzioni che sono disponibili solo quando il processore sta lavorando in modalità supervisore, istruzioni disponibili in modalità utente che permettono l'interfacciamento con programmi in modalità supervisore, e istruzioni per la manipolazione del byte del codice di condizione del registro di stato.
La Tabella 4.9 riassume le istruzioni di controllo del sistema.

COMUNICAZIONI MULTITASK/MULTIPROCESSORE

In un sistema multitasking o in un sistema multiprocessore, un task o un processore qualsiasi devono essere in grado di escludere un altro task o un altro processore: senza questa possibilità un programma avrebbe bisogno

Tabella 4.9 Istruzioni di controllo del sistema.

Mnemonico	Operazione
ANDI	AND immediato con il registro di stato/il registro dei codici di condizione
BKPT	Breakpoint Trap
CHK	Trap sull'operando superiore fuori dai limiti
CHK2*	Trap su un operando fuori dai limiti
EORI	OR esclusivo immediato con il registro di stato
ILLEGAL	Trap di istruzione illegale
MOVE	Sposta da/verso registro di stato/registro dei codici di condizione
MOVEC+	Sposta da/verso registro di controllo
MOVES+	Sposta da/verso lo spazio di indirizzamento
RESET+	Attiva la linea RESET
STOP+	Arresta il processore
TRAP	Trap incondizionata
TRAPcc*	Trap condizionata
TRAPV	Trap sull'overflow
+ istruzione privilegiata *solo 68020	

di parecchia logica aggiuntiva per impedire a un altro programma di manipolare le stesse locazioni di memoria che esso stava utilizzando. I processori della famiglia 68000 permettono di utilizzare cicli indivisibili di "lettura-modifica-scrittura" che impediscono al processore di cedere il bus durante l'esecuzione di un'istruzione.

In questo gruppo di istruzioni troviamo anche le istruzioni che permettono l'interfacciamento con il coprocessore: un coprocessore può essere un'unità in grado di gestire via hardware le operazioni in virgola mobile, un'unità di gestione della memoria, un'unità di controllo dell'I/O o qualcosa di simile. Il 68020 è l'unico membro della famiglia 68000 che include nell'hardware l'interfaccia con il processore. L'Appendice B tratta in dettaglio l'interfaccia con il coprocessore.

La Tabella 4.10 riassume le istruzioni multitask/multiprocessore.

Tabella 4.10 Istruzioni multitask/multiprocessore

Mnemonico	Operazione
CAS*	(Compare and Swap) Confronto e scambio con l'operando
CAS2*	Confronto e scambio con operandi
cpBcc*	(Branch on coprocessor condition) Diramazione a seconda di una particolare configurazione del coprocessore
cpDBcc*	Test sul coprocessore, decremento e diramazione
cpGEN*	Istruzione generale del coprocessore
cpRESTORE*	Ripristina lo stato del coprocessore
cpSAVE*	Salva lo stato del coprocessore
cpScc*	Test su una condizione del coprocessore
cpTRAPcc*	Trap condizionata del coprocessore
TAS	Test and set su un operando

*solo 68020

4.2 Istruzioni cicliche, di prefetch, di pipelining e di caching

PIPELINING

La famiglia di processori 68000 implementa vari livelli di "pipelining". Il pipelining comporta l'esecuzione concorrente delle fasi di fetching (prelevamento) e di esecuzione delle istruzioni; grazie a questa sovrapposizione il processore può eseguire le istruzioni molto più velocemente rispetto ai processori che effettuano la fase di fetch e di esecuzione serialmente.
Il 68000, il 68008, il 68010 e il 68012 usano un prefetch di due word. Il processore comincia prelevando la word relativa all'istruzione: come incomincia la fase di decodifica, il processore preleva la word seguente dalla memoria di modo che, quando incomincia l'esecuzione dell'istruzione, sono già state prelevate dalla memoria due word relative all'istruzione. Come il processore usa le due word prelevate, esso preleva altre due word dalla memoria. Notate che, se l'istruzione effettua un branch, la parola che segue l'istruzione (già memorizzata internamente al processore) viene scartata. In ogni caso durante l'esecuzione il vantaggio derivante dal prefetch è molto maggiore rispetto alla perdita di tempo dovuta ad un prelevamento inutile.
Il 68020 opera in modo simile, tranne per il fatto che il prefetch avviene su tre word di una singola istruzione o su tre consecutive istruzioni composte da una singola word.

ISTRUZIONI CICLICHE

Il 68010 e il 68012 migliorano l'idea del pipelining: dal momento che il processore effettua il prefetch di due word, la Motorola aggiunge un controllo per vedere se l'istruzione prelevata è un DBcc. In certi casi il processore può eseguire l'istruzione nel ciclo senza effettuare un altro prelevamento.
L'istruzione DBcc ha tre operandi, un contatore di ciclo, una condizione di salto e uno spiazzamento relativo al branch. In particolari situazioni in cui un'istruzione lunga una word è seguita da un'istruzione DBcc che provoca il ritorno a questa istruzione, il processore non ha bisogno di prelevare alcuna istruzione dalla memoria mentre è all'interno del ciclo.
La Tabella 4.11 elenca le istruzioni cicliche.

Tabella 4.11 Istruzioni cicliche.

ABCD	CMPA	ROL
ADD	EOR	ROR
ADDA	LSL	ROXL
ADDX	LSR	ROXR
AND	MOVE	SBCD
ASL	NBCD	SUB
ASR	NEG	SUBA
CLR	NEGX	SUBX
CMP	NOT	TST
	OR	

ISTRUZIONI PER LA CACHE

Il 68020 oltre al pipelining, possiede una memoria cache di 256 byte integrata sul chip. Il processore copia le istruzioni nella cache dopo averle prelevate dalla memoria e mantiene traccia di quali indirizzi (istruzioni) ha nella cache in modo che, prima di effettuare un prelevamento da memoria, esso può prelevare l'istruzione direttamente dalla cache.
Ciò accelera la velocità di esecuzione del processore dal momento che esso non ha bisogno di aspettare un ciclo di memoria per leggere l'istruzione. In un ambiente multiprocessore, questo espediente può accelerare tutti i processori dal momento che il bus non è impegnato nella fase di fetching.
Notate che per motivi di efficienza, il processore non inserisce gli operandi nella cache; oltre a ciò un programma supervisore può scegliere di disabilitare la cache: ciò potrebbe risultare utile durante una routine di exception, in quanto un programma utente potrebbe trovarsi fuori dalla cache al momento della exception.

5

Descrizione dei segnali

In questo capitolo tratteremo i segnali di input e di output richiesti dalla famiglia 68000. Dal momento che i membri della famiglia sono piuttosto simili tra loro, tratteremo, per ragioni di chiarezza, prima i membri della famiglia che usano un bus dati a 16 bit (68000, 68010 e 68012) e successivamente il 68020 che è l'unico membro della famiglia ad usare un bus dati di 32 bit. Poiché la Motorola e gli altri produttori di chip 68000 offrono una vasta gamma di package, non tenteremo in questa sede di individuare le relazioni biunivoche piedino/segnale, rimandando il lettore all'Appendice D.

5.1 Segnali del 68000, 68008, 68010 e 68012

Le Figure da 5.1 a 5.4 mostrano i gruppi funzionali di segnali su ciascuno dei processori a 16 bit.

BUS DATI

D0-D15 è il bus dati bidirezionale (sul 68000 questo bus è D0-D7); A1-A23 è l'indirizzo del bus di output (sul 68008 è A0-A19 e sul 68012 è A1-A29 e A-31). A causa del packaging, i processori 68000 non richiedono il multiplexing degli indirizzi e delle linee dati; per quanto riguarda questa caratteristica essi differiscono da altri processori quali ad esempio l'Intel 8086, lo Z8000 e l'NS32000.

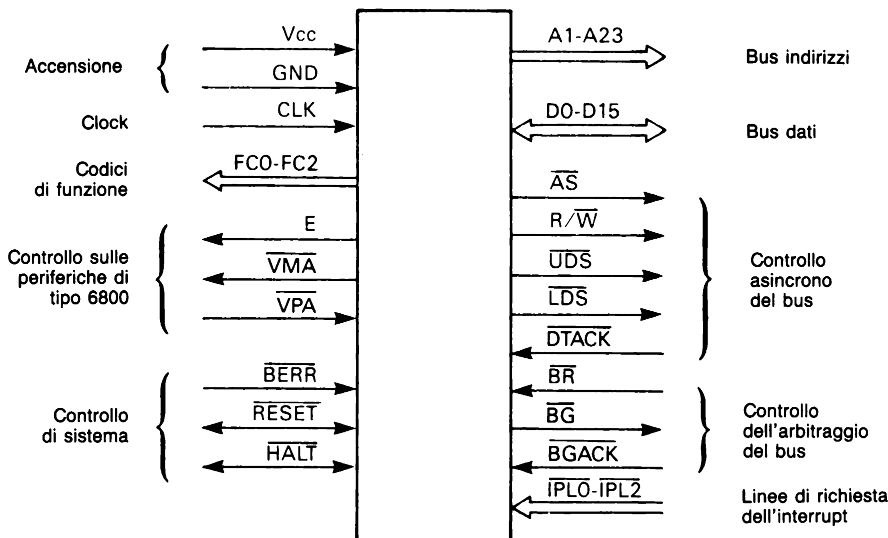


Figura 5.1 Segnali funzionali del 68000.

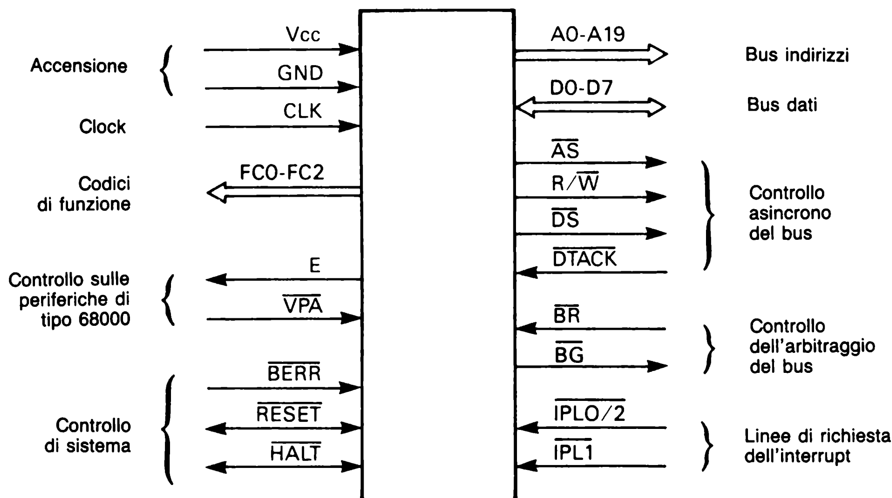


Figura 5.2 Segnali funzionali del 68008.

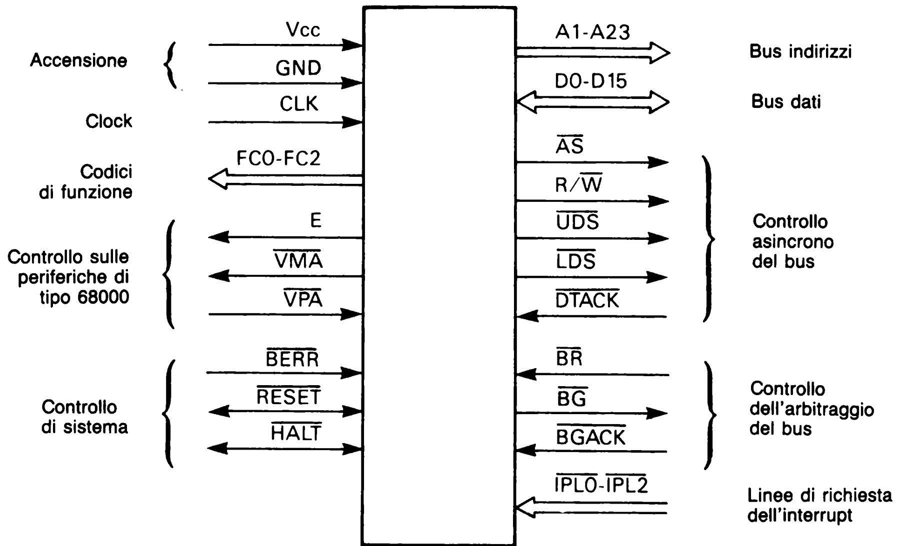


Figura 5.3 Segnali funzionali del 68010.

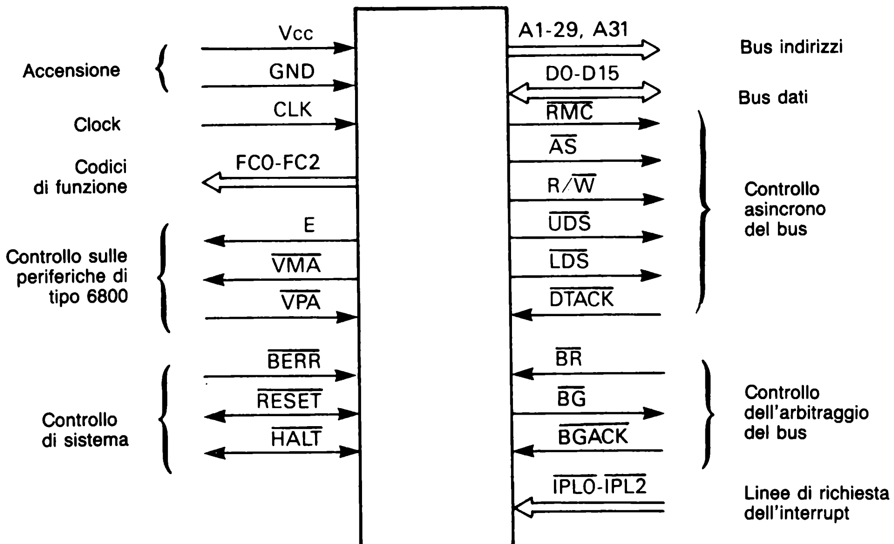


Figura 5.4 Segnali funzionali del 68012.

Notate che il 68008 è l'unico processore che include la linea con l'indirizzo meno significativo (A0). Il 68008 può accedere solo a 8 bit (un byte) di dati alla volta dal momento che il bus dati è "largo" solo 8 bit.

STROBE DATI

Il segnale di strobe dati ($\overline{DS}$) indica che il bus dati del 68008 è attivo.

STROBE DATI SUPERIORE/INFERIORE

Gli altri processori usano lo strobe dati superiore ($\overline{UDS}$) e quello inferiore ($\overline{LDS}$) per determinare se il dato deve essere trasferito sul byte superiore (byte più significativo), su quello inferiore (byte meno significativo) o su entrambi i byte del bus dati di 16 bit. La Tabella 5.1 definisce il significato degli $\overline{UDS}$ e degli $\overline{LDS}$ in relazione al bus dati.
Per accedere a un byte di dati pari, il processore attiva $\overline{UDS}$ e nega $\overline{LDS}$; il trasferimento dei dati avviene sui bit 8-15 del bus dati. Per accedere a un byte di dati dispari, il processore attiva $\overline{LDS}$ e nega $\overline{UDS}$. Per accedere all'intera word, il processore attiva sia $\overline{LDS}$ che $\overline{UDS}$: il trasferimento dei dati avviene su tutti i 16 bit del bus dati.
Questo schema di codifica contiene però una restrizione, e cioè che gli accessi alle word dati avvengono solo sugli indirizzi pari.
Il segnale relativo allo strobe indirizzi ($\overline{AS}$) è un output dal gestore dello strobe che indica che il bus degli indirizzi e lo strobe dati contengono dati validi.

Tabella 5.1 $\overline{UDS}$ e $\overline{LDS}$ sul 68000, 68010 e 68012.

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8-D15	D0-D7	Operazione
Alto	Alto				
Basso	Basso	Alto	Bit dati 8-15	Bit dati 0-7	Lettura di una word
Alto	Basso	Alto		Bit dati 0-7	Lettura di un byte
Basso	Alto	Alto	Bit dati 8-15		Lettura di un byte
Basso	Basso	Basso	Bit dati 8-15	Bit dati 0-7	Scrittura di una word
Alto	Basso	Basso	Bit dati 0-7	Bit dati 0-7	Scrittura di un byte
Basso	Alto	Basso	Bit dati 8-15	Bit dati 8-15	Scrittura di un byte

(Casella retinata) Nessun input od output valido

LETTURA/SCRITTURA

Il segnale di lettura/scrittura (R/$\overline{W}$) permette di stabilire se il trasferimento di dati dal bus è una lettura o una scrittura.

CONFERMA DEL TRASFERIMENTO DEI DATI

$\overline{DTACK}$ (Data Transfer Acknowledge) è il segnale di input che conferma l'avvenuto trasferimento di dati: la logica esterna (ad esempio la memoria) deve attivare $\overline{DTACK}$ per segnalare al processore l'avvenuta ricezione dei dati sul bus (in caso di scrittura) o per segnalare che i dati richiesti dal processore sono stati piazzati sul bus dati (in caso di lettura). Se necessario, il processore inserirà automaticamente nel proprio ciclo dei cicli di wait fino al momento in cui non riceverà $\overline{DTACK}$. Questo meccanismo rende possibile il dialogo con periferiche di velocità differente.

CODICI DI FUNZIONE

FC0, FC1 e FC2 sono gli output dei codici di funzione e rappresentano lo stato del sistema. Questi output (validi solo quando viene attivato $\overline{AS}$) identificano, per quanto riguarda la famiglia 68000, il tipo di attività che il bus sta svolgendo: la Tabella 5.2 riassume tutto ciò. Come potete vedere dalla tabel-

Tabella 5.2 Tabella dei codici di funzione.

FC2	FC1	FC0	Tipo di ciclo macchina
0	0	0	(riservato)
0	0	1	Accesso alla memoria dati utente
0	1	0	Accesso alla memoria di programma dell'utente
0	1	1	(riservato)
1	0	0	(riservato)
1	0	1	Accesso ai dati di supervisore
1	1	0	Accesso al programma supervisore
1	1	1	Spazio di CPU (ricezione dell'interrupt)

(retinato) Riservato, temporaneamente indefinito

la, i codici di funzione di output separano gli accessi in memoria a seconda che l'accesso venga effettuato dal programma o dai dati, dall'utente o dal supervisore. Ciò permette a una unità di gestione della memoria (quale ad esempio il 68451) di controllare l'accesso a zone di memoria protette e non protette.
Una quinta codifica dei codici di funzione è definita per lo "spazio di CPU". Notate che la documentazione del 68000 e del 68008 si riferisce a questo codice come "riconoscimento di interrupt". In generale il processore usa lo spazio di CPU per comunicazioni con le periferiche che non siano le normali operazioni di lettura e di scrittura: un esempio di questo tipo di operazioni può essere quello del riconoscimento di un interrupt (come vedremo più tardi).
Il 68010 e il 68012 permettono a un programma di definire il codice di funzione di output per una istruzione di MOVE attraverso i registri del codice di funzione (SPC e DFC).

SEGNALI DI INTERRUPT

Quando una periferica vuole interrompere il processore (ad esempio per indicare la presenza di una nuova word di dati), essa attiva una o più linee di interrupt $\overline{\text{IPL0}}$, $\overline{\text{IPL1}}$ e $\overline{\text{IPL2}}$. Questi tre input rappresentano il valore binario del livello di interrupt richiesto.
Ricorderete da quanto detto precedentemente che il registro di stato contiene una maschera di interrupt di 3 bit che determina i livelli di interrupt che possono iniziare la gestione delle exception. Per attivare un interrupt, una periferica attiva il proprio livello di priorità sulle linee di interrupt. Se la configurazione della maschera di interrupt corrisponde, la maschera stessa provvederà a ricevere l'interrupt attraverso le linee dei codici di funzione. La gestione delle exception verrà trattata nel Capitolo 7.
Il 68008 combina $\overline{\text{IPL0}}$ e $\overline{\text{IPL2}}$ in un unico segnale. Ciò limita questo processore ai livelli di interrupt 0, 2, 5 e 7.

ERRORI SUL BUS

$\overline{\text{BERR}}$ è il segnale di input relativo agli errori che si sono verificati sul bus. Quando questo segnale viene attivato, il 68000 inizia un processo di exception. Lo scopo di $\overline{\text{BERR}}$ è quello di informare il processore che un dispositivo esterno non ha risposto a un'operazione di lettura o di scrittura (non ha attivato cioè $\overline{\text{DTACK}}$). $\overline{\text{BERR}}$ può essere usata anche da una unità di gestione della memoria (MMU) per indicare che il processore ha tentato l'accesso a una zona di memoria protetta.
Nel primo caso, per generare $\overline{\text{BERR}}$ è necessario che il sistema disponga di

hardware esterno in grado di generare il segnale BERR. Nel secondo caso invece la linea di input BERR verrà solitamente connessa al page fault o a una linea equivalente di MMU.
BERR produce risultati leggermente differenti sui processori in grado di supportare memoria virtuale, e cioè il 68010 e il 68012. Il Capitolo 7 descriverà in dettaglio queste, seppur lievi, differenze.

HALT DEL PROCESSORE

Il segnale di HALT svolge parecchie funzioni; esso è un segnale sia di input che di output e può lavorare da solo o in congiunzione con altre linee. Quando viene attivato come un'unica linea di input, HALT forza il processore in uno stato inattivo, stato in cui rimarrà fino a quando il segnale di HALT non verrà negato.
HALT può funzionare anche come segnale di output indicante che il processore ha cessato l'esecuzione delle istruzioni, ad esempio a causa di una condizione di bus fault. La logica esterna può quindi riconoscere questa condizione.
HALT e BERR possono essere usati insieme: quando HALT viene attivato unitamente a BERR, il processore riesegue l'ultimo ciclo.

RESET

RESET, come HALT, è un segnale bidirezionale. Se il processore esegue un'istruzione di RESET, esso attiva questa linea. RESET può funzionare anche come linea di input quando viene usato in congiunzione con la linea di HALT. Quando questi segnali vengono attivati insieme, il 68000 viene inizializzato (includendo anche una trappola per i valori di reset).

CLOCK

Per funzionare il processore ha bisogno di un segnale di temporizzazione. Il segnale di clock, CLK, fornisce questo valore.

SEGNALI DI ARBITRAGGIO DEL BUS

I segnali di richiesta del bus (BR o Bus Request), di cessione del bus (BG o Bus Grant) e di ricezione del segnale di cessione del bus (BGACK o Bus Grant Acknowledge) sono segnali di arbitraggio del bus. Questi segnali sono usati in sistemi in cui altri dispositivi, quali ad esempio i DMA, possono funziona-

re come master e richiedere quindi il controllo del bus. I dispositivi esterni richiedono il controllo del bus di sistema attivando il segnale $\overline{BR}$. Dopo aver completato il ciclo di bus corrente, il processore rilascia il bus.
Il processore manda in output il segnale $\overline{BG}$ per comunicare al dispositivo che effettua la richiesta che il bus sarà disponibile alla fine del ciclo di bus corrente. Il dispositivo comunica al 68000 la ricezione del messaggio di cessione del bus attivando il segnale $\overline{BGACK}$.
La descrizione dei meccanismi di cessione del bus è molto più complesso di così: entreremo nei dettagli relativi all'handshaking necessario nel Capitolo 6.

SEGNALI DELLA FAMIGLIA 6800

Dal momento che la famiglia 68000 è un lontano parente della serie di processori 6800, Motorola ha scelto di includere tre segnali di bus di modo che i progettisti di sistema potessero costruire sistemi in grado di usare i chip 6800. Questi segnali sono il segnale di abilitazione (E, Enable), il segnale di accesso valido a periferica ($\overline{VPA}$ o Valid Peripheral Address) e di indirizzo valido di memoria ($\overline{VMA}$ o Valid Memory Access).
A differenza del 68000, il 6800 usa un'interfaccia di bus sincrona invece che un'interfaccia asincrona. Ciò significa che il trasferimento di dati attraverso il sistema è sincronizzato tramite un segnale di clock. Il segnale E mandato in output dal 68000 implementa questo orologio. La frequenza di E è uguale a un decimo del segnale CLK in input al 68000; il periodo di E è uguale a 10 periodi di CLK: E è basso per sei cicli CLK e alto per 4 cicli CLK.
Il segnale $\overline{VPA}$ è usato dai dispositivi di sistema realizzati con la famiglia 6800 per informare il 68000 che è necessario effettuare un trasferimento di dati di tipo 6800. Il sistema deve fornire la logica esterna in grado di determinare quali indirizzi corrispondono a periferiche della famiglia 6800. Questa logica deve attivare il segnale $\overline{VPA}$.
Quando il 68000 riceve il segnale $\overline{VPA}$, esso altera il proprio meccanismo relativo alla temporizzazione del trasferimento in modo da sincronizzarsi con il segnale E; a questo punto il processore attiva il segnale $\overline{VMA}$ e il trasferimento dei dati ha inizio.
Il 68008 non implementa il segnale $\overline{VMA}$.
L'interfacciamento con un membro della famiglia 6800 è trattato in dettaglio nell'Appendice B.

5.2 Segnali del 68020

Per le sue caratteristiche particolarmente sofisticate, il 68020 richiede alcune aggiunte e alcune modifiche al set dei segnali di input e di output. Esami-

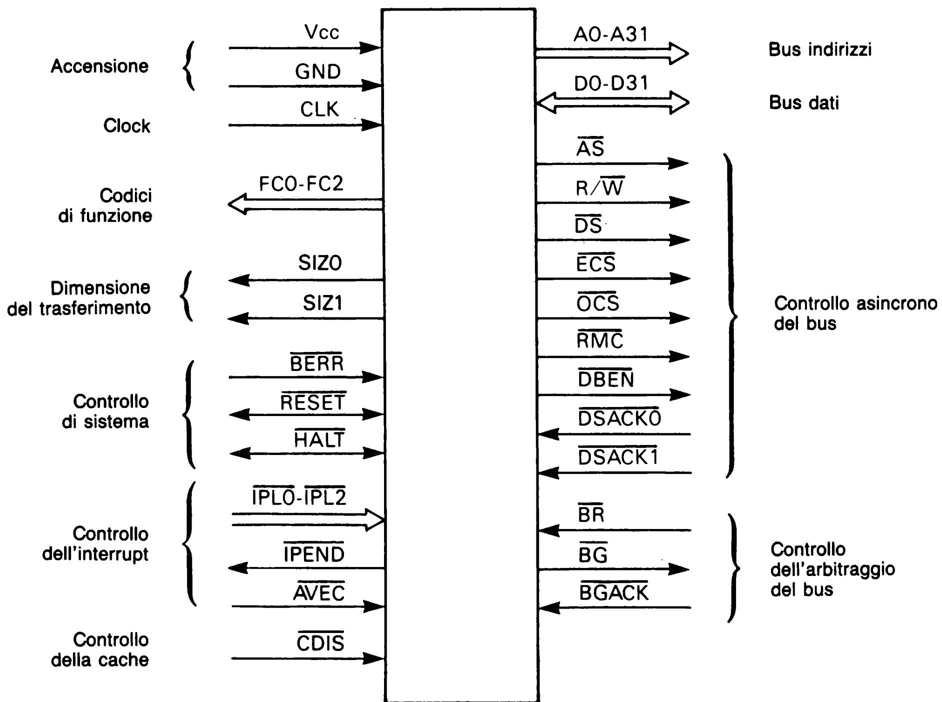


Figura 5.5 Segnali funzionali del 68020.

nando i segnali del 68020 troverete che i nomi e le funzioni di numerosi segnali sono molto simili a quelli degli altri elementi della famiglia; le differenze stanno nel fatto che sia il bus dati che quello indirizzi sono a 32 bit e che in questo processore mancano le linee caratteristiche della famiglia 6800.

La Figura 5.5 illustra modo in cui possono essere raggruppati, a seconda della loro funzione, i segnali del 68020; l'Appendice D fornisce le corrispondenze segnale-piedino.

BUS DATI

Il bus dati bidirezionale è largo 32 bit (D0-D31). Il 68020 è in grado di dimensionare il bus dinamicamente: ciò significa che esso può passare dati formati da 1, 2, 3 o 4 byte usando una qualsiasi parte del bus dati. Nel Capitolo 6 tratteremo più approfonditamente la dimensionabilità del bus.

BUS INDIRIZZI

Il bus indirizzi è un bus di 32 bit (A0-A31). Da rimarcare è la presenza di A0 che, come già visto, non viene utilizzato in alcuni modelli precedenti. La larghezza del bus permette di indirizzare uno spazio di memoria di 4 gigabyte.

DIMENSIONI DEI DATI DA TRASFERIRE

I mezzi per controllare la larghezza del bus dati necessaria per effettuare un particolare trasferimento di dati sono rappresentati dai segnali SIZ0 e SIZ1. Questi segnali si combinano insieme per fornire il numero di byte coinvolti in un'operazione di trasferimento. Nel Capitolo 6 entreremo nei dettagli circa l'uso dei segnali di dimensionamento in congiunzione con i segnali di trasferimento.

LETTURA/SCRITTURA

Come gli altri membri della famiglia, il 68020 usa un certo numero di segnali di controllo del bus per gestire l'interfaccia sincrona con le periferiche. Il segnale di lettura/scrittura (Read/Write o $R/\overline{W}$) indica la direzione del trasferimento. Notate che questa linea ha due stati true: true alto significa lettura e true basso significa scrittura.

STROBE DATI

Lo strobe dati ($\overline{DS}$) indica che il bus dati è attivo. Se viene attivato durante un ciclo di lettura, $\overline{DS}$ indica che la periferica deve caricare bus dati, mentre se viene attivato durante un ciclo di lettura esso indica che la periferica può appropriarsi dei dati sul bus.

STROBE INDIRIZZI

Lo strobe indirizzi (Address Strobe o $\overline{AS}$) è un segnale di output che indica che il processore ha mandato un codice di funzione, un indirizzo, una dimensione o una informazione circa lo stato di un'operazione di $R/\overline{W}$ sulle linee appropriate.

CODICI DI FUNZIONE

I segnali dei codici di funzione (FC0, FC1 e FC2) identificano lo stato corrente del processore e lo spazio di indirizzamento richiesto. FC0-FC2 vengono combinati (come mostrato nella Figura 5.2) per identificare lo spazio di indirizzamento di dati o programmi (codice) e lo spazio di indirizzamento riservato all'utente o al supervisore.

Una quinta combinazione identifica lo "spazio di CPU". In questo spazio sono inclusi quattro sottotipi: riconoscimento di breakpoint, operazioni di modulo, comunicazioni col coprocessore e riconoscimento di interrupt. Il processore usa il bus indirizzi sia per distinguere fra queste modalità che per passare informazioni alle periferiche. La Figura 5.6 mostra la codifica dello spazio di CPU.

Notate che una routine supervisore può richiedere uno spazio di indirizzamento particolare per tutte le istruzioni MOVES fornendo il valore del codice di funzione ai registri sorgente e destinazione (SFC e DFC).

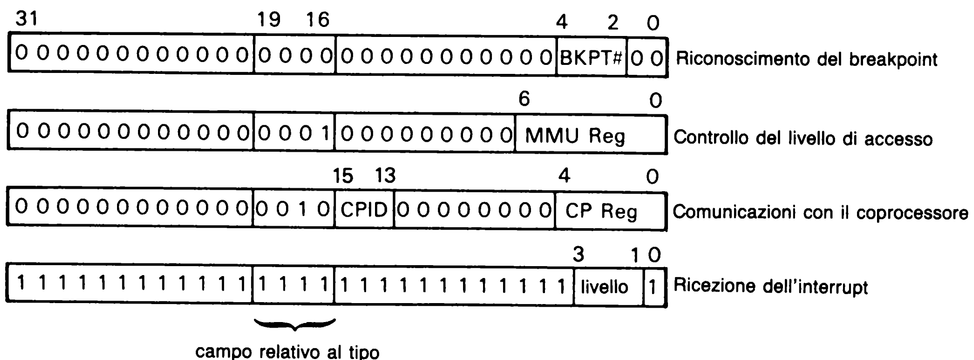


Figura 5.6 Codifica del bus indirizzi relativo allo spazio di CPU.

TRASFERIMENTO DATI E ACQUISIZIONE DELLA DIMENSIONE DEL DATO

I segnali di trasferimento dati e di acquisizione della dimensione dei dati di input ($\overline{DSACK0}$ e $\overline{DSACK1}$) indicano al processore che è stato completato un trasferimento dati. Se si tratta di un ciclo di lettura ciò significa che la periferica ha posto il dato sul bus e che il processore deve incamerarlo. Se si tratta di un ciclo di scrittura ciò significa che la periferica ha letto il dato e che quindi il processore può riprendere a lavorare. Il Capitolo 6 fornisce ulteriori informazioni circa l'handshaking fra il processore e le periferiche e la necessità di due linee di ACK anziché una.

PARTENZA DEL CICLO

Per controllare al meglio le comunicazioni fra il processore e le periferiche, il 68020 include due cicli di partenza dei segnali di output. Il processore attiva il segnale di inizio del ciclo esterno ($\overline{\text{ECS}}$) durante la prima metà del periodo di clock di ogni ciclo di bus. Il ciclo del segnale di partenza dell'operando funziona allo stesso modo, tranne per il fatto che il processore attiva il segnale solo durante il primo ciclo di bus relativo al trasferimento di un operando.

SEGNALE DI LETTURA-MODIFICA-SCRITTURA

Il segnale di output di lettura-modifica-scrittura ($\overline{\text{RMC}}$) indica che il bus viene escluso dal controllo esterno. Le uniche istruzioni del 68020 che forzano questo segnale sono quelle di Test and Set (TAS) e di confronto e scambio (CAS e CAS2). Questa possibilità di escludere il bus assicura l'integrità nei sistemi multiprocessore.

ABILITAZIONE BUFFER DATI

Il segnale di abilitazione del buffer dati ($\overline{\text{DBEN}}$) abilita i buffer dati esterni durante il trasferimento dei dati. Esso permette di modificare il segnale R/$\overline{\text{W}}$ senza pericolo di conflitti sul buffer esterno.

DISABILITAZIONE DELLA CACHE

Il segnale di disabilitazione della cache ($\overline{\text{CDIS}}$) disabilita la cache istruzione integrata sul chip.

RICHIESTA DI INTERRUPT

Il 68020 supporta sette livelli di interrupt. Le linee di richiesta di interrupt in input al processore stanno a indicare che una periferica ha bisogno di interrompere il processore. Al momento della ricezione della richiesta, il processore confronta queste linee con la maschera delle priorità di interrupt contenuta nel registro di stato. La gestione degli interrupt è trattata in dettaglio nel Capitolo 7.

INTERRUPT IN ATTESA DI SERVIZIO

Il segnale di interrupt in attesa di servizio (IPEND) che il processore manda in output indica che il processore sta ricevendo un livello di interrupt codificato (via IPL0-IPL2) che è più alto del livello di interrupt memorizzato nella maschera di interrupt del registro di stato, o che il processore ha ricevuto un interrupt non mascherabile.

AUTOVETTORE

Dopo che il processore ha riconosciuto un segnale di interrupt, il dispositivo che ha interrotto il processore normalmente restituisce un vettore che identifica l'elemento della tabella dei vettori in cui il processore può trovare la routine di interrupt appropriata. Il 68020 permette l'autovettorizzazione tramite la linea di input per l'autovettore (AVEC).
Quando AVEC viene attivato, il processore sceglierà automaticamente un vettore nella tabella invece di aspettare che il dispositivo specifichi un vettore. I processori 68000 precedenti usavano VPA per richiedere la funzione di autovettorizzazione. Nel Capitolo 7 tratteremo in dettaglio l'autovettorizzazione.

ERRORI SUL BUS

BERR è il segnale di input di errore sul bus. Quando questo segnale viene attivato, il 68020 dà inizio a un processo di exception. Lo scopo di BERR è quello di informare il processore che un dispositivo esterno non ha risposto a un'operazione di lettura o di scrittura (ovverosia non ha attivato DSACK0/DSACK1). BERR può essere usato da un'unità di gestione della memoria (MMU) per indicare che il processore ha tentato di accedere a una zona di memoria protetta.
Nel primo caso, per generare un BERR, il sistema deve poter usufruire di hardware aggiuntivo in grado di generare il segnale BERR. Nel secondo caso, la linea di input BERR verrà di solito connessa al page fault o a una linea equivalente del MMU.

HALT DEL PROCESSORE

Il segnale HALT svolge parecchie funzioni. Esso è sia un segnale di input che un segnale di output e può lavorare da solo o in congiunzione con altre linee. Quando viene attivato da solo come linea di input, HALT forza il processore in uno stato inattivo che rimarrà tale fino a che non verrà negato il segnale HALT.

HALT può funzionare anche come segnale di output indicante il fatto che il processore ha cessato l'esecuzione delle istruzioni, ad esempio a causa di una doppia condizione di errore sul bus (double bus fault): quest'ultima condizione può essere facilmente individuata dalla logica esterna.
HALT e BERR possono essere usate insieme; quando HALT viene attivato insieme a BERR, il processore riesegue l'ultimo ciclo di bus.

RESET

RESET, come HALT, è un segnale bidirezionale; se il processore esegue un'istruzione di RESET, viene attivata questa linea. RESET può anche funzionare come linea di input che, quando viene attivata, forza il 68020 ad eseguire il vettore di reset.

CLOCK

Per funzionare il processore richiede un segnale di temporizzazione: questo valore viene fornito dal segnale di input CLK.

SEGNALI DI ARBITRAGGIO DEL BUS

I segnali di richiesta del bus BR (Bus Request), di disponibilità a cedere il bus BG (Bus Grant) e di conferma della ricezione di quest'ultimo segnale BGACK (Bus Grant Acknowledge) sono segnali per l'arbitraggio del bus. Questi segnali sono usati in sistemi in cui altri dispositivi, quali ad esempio i controllori del DMA, possono funzionare come padroni del bus richiedendone il controllo. I dispositivi esterni richiedono l'accesso al bus di sistema attivando il segnale BR: il 68020 rilascerà il bus dopo aver completato il ciclo di bus corrente.
Il processore manda in output il segnale BG per comunicare al dispositivo che ha effettuato la richiesta che il bus risulterà disponibile alla fine del ciclo corrente. Il dispositivo riconoscerà quindi che il 68020 ha rinunciato al bus attivando il segnale BGACK; il segnale verrà mantenuto alto fino al momento in cui quel dispositivo non avrà finito di utilizzare il bus.
Notate che, nel cedere il controllo del bus, vi è molto più handshaking; questo aspetto verrà trattato più dettagliatamente nel Capitolo 6.

6

Temporizzazione e operazioni sul bus

Il meccanismo di base di temporizzazione per la famiglia 68000 è piuttosto semplice; l'esecuzione delle istruzioni consiste in una combinazione di cicli interni e di cicli di accesso al bus. Questo capitolo spiega i cicli di accesso al bus richiesti per introdurre i dati all'interno del sistema, per comunicarli alla logica esterna e per farli circolare all'interno del microprocessore stesso.
Tutti i membri della famiglia 68000 usano trasferimenti asincroni per trasferire i dati all'esterno del microprocessore: ciò significa che il processore e le periferiche usano linee di handshake per coordinare il trasferimento dei dati. Usando questo tipo di trasferimento, il processore permette di dialogare con periferiche con differenti velocità. Ad esempio molti sistemi 68000 usano memorie molto veloci (e molto costose) come memorie cache e una memoria più lenta (e meno costosa) da usare come RAM.
I processori eseguono due operazioni fondamentali: lettura e scrittura. Si ha un'operazione di lettura quando una periferica introduce un dato sul bus dedicato ai dati; si ha invece un'operazione di lettura quando la periferica accetta dei dati provenienti dal bus dati. Generalmente la sorgente o la destinazione di queste operazioni è un registro del processore, anche se l'istruzione MOVE permette di spostare i dati da una locazione di memoria ad un'altra.
Come già fatto nel capitolo precedente separeremo i processori il cui bus dati è di 8 o 16 bit (68000, 68008, 68010, 68012) dal 68020. Ciò viene fatto per due ragioni principali, ovvero per il fatto che il 68020 permette il dimensionamento dinamico e il disallineamento degli operandi.
Dopo la trattazione separata dei meccanismi di temporizzazione per le operazioni di lettura e scrittura, approfondiremo l'argomento delle caratteristiche comuni (stato di halt e di stop, reset del sistema e arbitraggio del bus) alla fine del capitolo.

6.1 Temporizzazione e operazioni sul bus per il 68000-68012

TEMPORIZZAZIONE DELLA LETTURA

La Figura 6.1 illustra il meccanismo di temporizzazione per un'operazione di lettura di una word. Si inizia suddividendo ogni periodo di clock in due stati etichettati nelle figure come S*n*, dove *n* è pari per la prima parte del ciclo e dispari per la seconda parte.
Nello stato 0 (S0) del ciclo di lettura i bus dati e indirizzi sono entrambi in uno stato ad alta impedenza: a questo punto il 68000 non sta usando il bus di sistema. Le informazioni circa gli indirizzi delle varie periferiche vengono mandate sul bus all'inizio dello stato 1 (S1), mentre i codici di funzione (FC0-FC2) contengono le informazioni circa il ciclo che il processore sta ese-

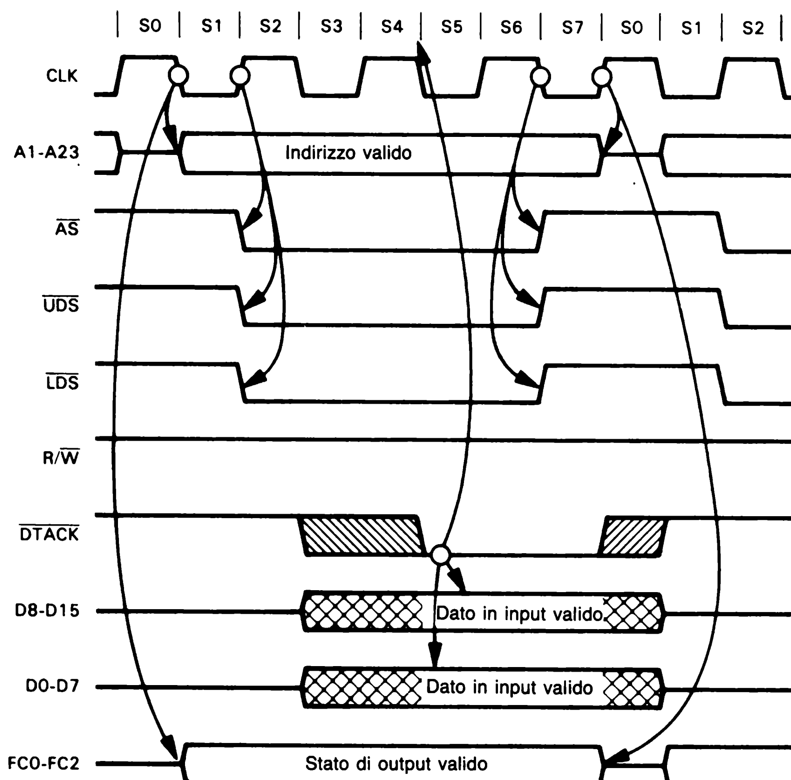


Figura 6.1 Temporizzazione della lettura di una word (68000, 68010, 68012).

guendo. Lo strobe indirizzi (AS) viene attivato all'inizio dello stato 2 e può essere usato dalla logica esterna per catturare le informazioni che si trovano sul bus indirizzi.
Simultaneamente lo strobe dati superiore (UDS o Upper Data Strobe) e lo strobe dati inferiore (LDS o Lower Data Strobe) vengono attivati per abilitare la selezione sia del byte più significativo sia di quello meno significativo della parola di 16 bit. Notate che questi segnali non sono dei veri e propri strobe, in quanto non vi è alcun dato pronto per essere letto o per essere scritto. È più corretto quindi pensare ad essi come segnali per selezionare all'interno di una word il byte più significativo e quello meno significativo. Il segnale R/W viene di solito attivato per evitare che questo output non cambi durante un ciclo di lettura.
A questo punto il 68000 attende che la periferica in questione presenti il proprio dato sul bus: dopo che ciò si è verificato la periferica attiva il segnale di ricezione dei dati DTACK al 68000. Quest'ultimo attende l'attivazione del segnale DTACK fino allo stato 5. Se a questo punto il segnale non è stato ancora attivato, il processore inserisce automaticamente degli "stati di wait" (SW o Wait State) nel ciclo di temporizzazione.
Quando la periferica attiva DTACK, il ciclo di lettura continua con lo stato 5; alla fine dello stato 6 i segnali AS, UDS e LDS vengono negati e il processore cattura i dati dal bus dati e li inserisce in un proprio registro interno. I dispositivi esterni possono usare la negazione di questi segnali come indicazione che il processore ha ricevuto il dato e che essi possono rimuovere i dati dal bus. Il 68000 mantiene le informazioni circa gli indirizzi e i codici di funzione fino alla fine dello stato 7 per permettere lo slittamento dei segnali all'interno del sistema.
Notate che quando i dispositivi esterni si accorgono che il 68000 ha catturato il dato dal bus dati (ricevendo la negazione di AS, UDS e LDS), la periferica deve negare DTACK immediatamente in modo da non interferire con l'inizio del ciclo di bus successivo.

STATI DI WAIT

Ripensando all'inserimento di stati di wait che può avvenire durante le operazioni di lettura, come illustrato in Figura 6.1, noterete che esso può avvenire fra gli stati 4 e 5. In questo stato il 68000 manterrà un indirizzo valido sul bus indirizzi e continuerà ad attivare AS, UDS e LDS fino a che la periferica non asserisce DTACK. Notate che vi sarà sempre un numero pari di stati di wait, in quanto le operazioni del 68000 sono basate sul clock e, per ogni ciclo di clock, vi sono due stati.

TEMPORIZZAZIONE DELLE OPERAZIONI DI LETTURA DI UN BYTE

La temporizzzazione per le operazioni di lettura di un byte sono illustrate nella Figura 6.2: questa figura mostra la lettura di un byte di dati pari e un byte dispari. Come potete notare, le uniche differenze fra la temporizzazione di questa operazione e la temporizzazione dell'operazione di lettura di una word mostrata in Figura 6.1 stanno nel fatto che in questo caso viene attivato solo uno dei segnali $\overline{UDS}$ e $\overline{LDS}$ e che vengono utilizzate solo 8 linee del bus dati. Per la lettura di un byte ad un indirizzo pari vengono attivati il segnale $\overline{UDS}$ e le linee dati D8-D15. Per la lettura di un byte ad un indirizzo dispari viene invece attivato LDS e i dati vengono posti sulle linee D0-D7.

Nota: Non lasciatevi ingannare dalla Figura 6.2 deducendo che il 68000 legge sempre due byte consecutivi. Abbiamo semplicemente mostrato le due operazioni di lettura consecutivamente per poter descrivere il meccanismo di temporizzazione di entrambe.)

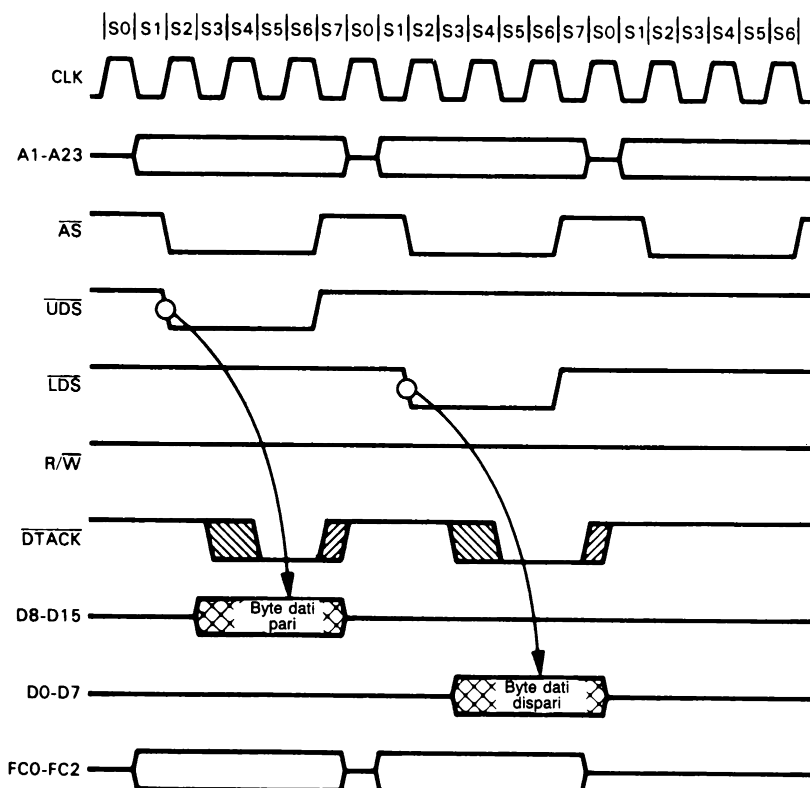


Figura 6.2 Temporizzazione della lettura di un byte (68000, 68010, 68012).

TEMPORIZZAZIONE DELLA LETTURA PER IL 68008

La temporizzazione dell'operazione di lettura per il 68008 è identica a quella degli altri membri della famiglia con la differenza che esso possiede solo uno strobe dati ($\overline{DS}$) e che il bus dati possiede solo 8 linee (D0-D7). La Figura 6.3 mostra il ciclo di lettura del 68008.

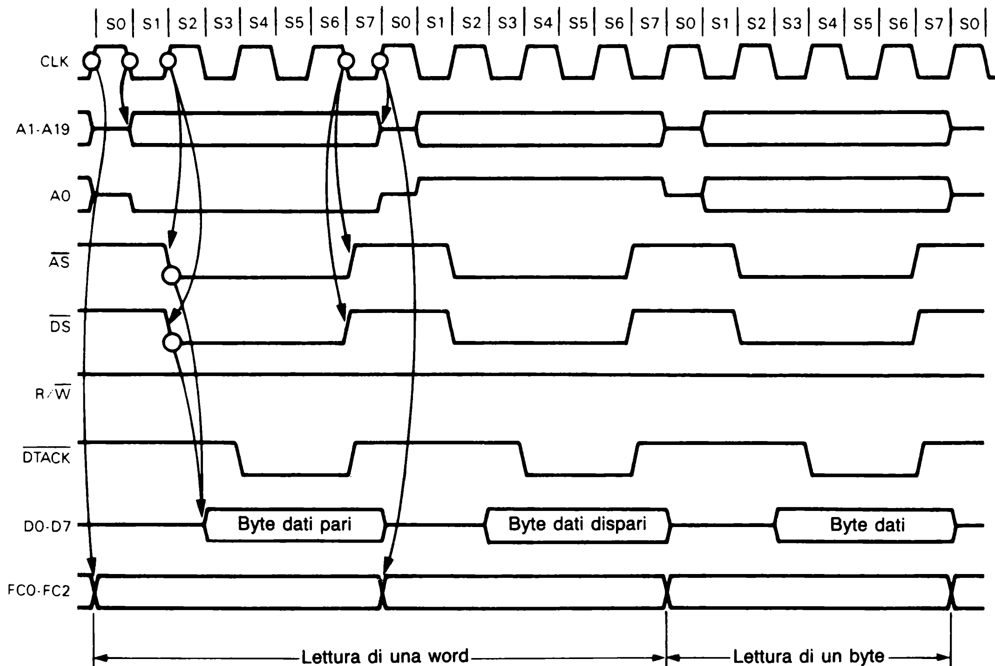


Figura 6.3 Temporizzazione della lettura di una word e della lettura di un byte (68008).

TEMPORIZZAZIONE DELLA SCRITTURA

La temporizzazione delle operazioni di scrittura di una word è illustrata nella Figura 6.4. Come nel caso delle operazioni di lettura, l'indirizzo della periferica (memoria, dispositivo di I/O e così via) viene emesso all'inizio dello stato 1 unitamente al codice di funzione appropriato. Se il bus dati è stato utilizzato dal 68000 nel ciclo precedente, il processore riporta, durante lo stato 1, tutti i dati in output allo stato ad alta impedenza, attiva lo strobe indirizzi ($\overline{AS}$) e abbassa il segnale di lettura/scrittura per indicare un ciclo di scrittura.

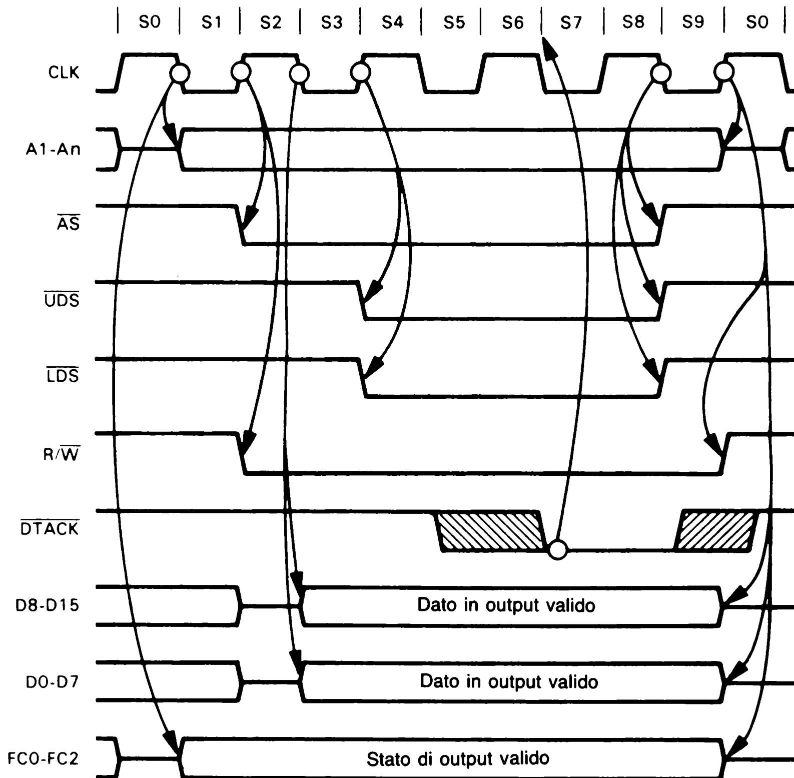


Figura 6.4 Temporizzazione della scrittura di una word (68000, 68010, 68012).

Ancora una volta $\overline{AS}$ può essere usato per catturare l'indirizzo, mentre il segnale R/$\overline{W}$ indica alla periferica che il 68000 sta piazzando il dato sul bus dati. Fino al momento in cui il 68000 non inserisce il dato sul bus, all'inizio dello stato 3, non vengono più attivati segnali. I segnali $\overline{UDS}$ e $\overline{LDS}$ vengono attivati all'inizio dello stato 4. Durante le operazioni di scrittura questi segnali possono essere usati come segnali di strobe positivi in quanto indicano che il dato che si trova sul bus è valido.

STATI DI WAIT

La periferica deve rispondere agli strobe dati attivando il segnale di ricezione del trasferimento dati: se ciò avviene entro l'inizio dello stato 7, il processore continuerà a lavorare ininterrottamente. Se $\overline{DTACK}$ non viene attivato entro l'inizio dello stato 7, il 68000 inserisce automaticamente stati di wait

all'interno del ciclo di scrittura. Questo meccanismo funziona esattamente come l'inserimento di stati di wait all'interno di un ciclo di lettura tranne per il fatto che l'inserimento avviene in uno stato differente del ciclo.
Il 68000 emette i dati sulle linee D0-D15 attraverso l'operazione di scrittura. Lo strobe indirizzi ($\overline{AS}$) e gli strobe dati ($\overline{LDS}$ e $\overline{UDS}$) vengono negati all'inizio dello stato 9, e il segnale R/$\overline{W}$ viene alzato alla fine dello stato 9. A questo punto il bus indirizzi, il bus dati e l'output dei codici di funzione vengono riportati al loro stato ad alta impedenza; in questo modo il bus di sistema viene liberato per altre operazioni. La periferica deve negare il segnale $\overline{DTACK}$ dopo aver riconosciuto la negazione dei segnali di strobe dati e indirizzi: ciò assicura il regolare svolgimento delle operazioni successive sul bus.

TEMPORIZZAZIONE DELLE OPERAZIONI DI SCRITTURA DI UN BYTE

La temporizzazione delle operazioni di scrittura di un byte è illustrata nella Figura 6.5; come potete notare, l'unica differenza fra questa operazione e quella di scrittura di una word è che durante la scrittura vengono attivati il segnale UDS o il segnale LDS.

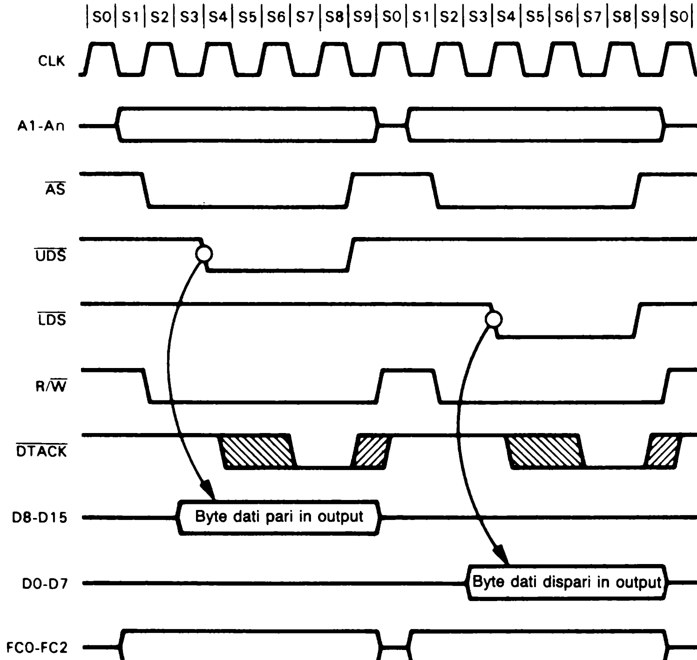


Figura 6.5 Temporizzazione della scrittura di un byte (68000, 68010, 68012).

TEMPORIZZAZIONE DELLE OPERAZIONI DI SCRITTURA DEL 68008

La temporizzazione dell'operazione di scrittura per il 68000 è identica a quella degli altri membri della famiglia con la differenza che esso possiede solo uno strobe dati (DS) e che il bus dati possiede solo 8 linee (D0-D7). La Figura 6.6 mostra il ciclo di scrittura del 68008.

TEMPORIZZAZIONE DEL CICLO LETTURA-MODIFICA-SCRITTURA

Il ciclo di lettura-modifica-scrittura fornito dal 68000 è piuttosto diverso da quello supportato dagli altri microprocessori. Il 68000 usa questo ciclo solo durante l'esecuzione dell'istruzione di Test and Set (TAS); questa istruzione legge un byte di dati, setta i codici di condizione in conformità con il contenuto di quel byte, setta il bit 7 di quel byte e quindi lo riscrive in memoria. L'istruzione TAS viene sfruttata per fornire un mezzo sicuro per far comunicare i vari microprocessori in un sistema multiprocessing. Il ciclo di lettura-modifica-scrittura non può essere interrotto dalla richiesta di un altro processore master di poter usare il bus; in pratica quindi i dati cui si accede

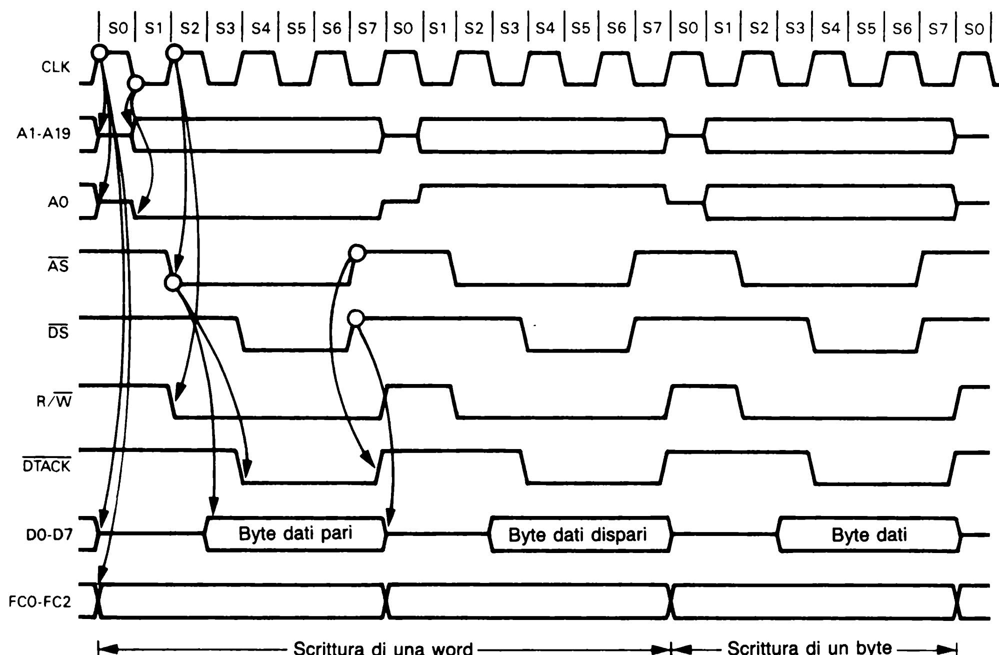


Figura 6.6 Temporizzazione della scrittura di un byte e di una word (68008).

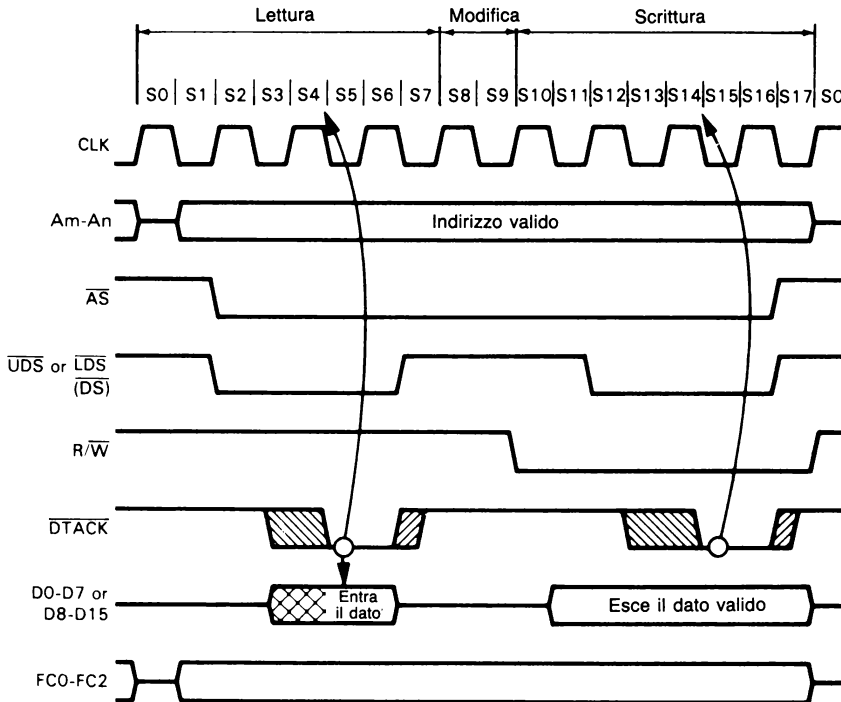


Figura 6.7 Temporizzazione del ciclo lettura-modifica-scrittura (68000-68012).

attraverso l'istruzione TAS non possono essere usati da un altro processore master collegato al bus durante tutta la durata dell'istruzione.
La Figura 6.7 mostra la temporizzazione del ciclo di lettura-modifica-scrittura.

6.2 Temporizzazione e operazioni sul bus per il 68020

Come già accennato nel precedente capitolo, la temporizzazione e le operazioni di bus del 68020 differiscono da quelle dei suoi predecessori in due aree principali, ovvero quelle della dimensionabilità dinamica del bus e del disallineamento. Il 68020 possiede anche operazioni di CPU più elaborate e alcuni nuovi segnali in grado di connettere il processore al bus di sistema. Dal momento che le operazioni di lettura e quelle di scrittura sono influenzate notevolmente dalle dimensioni e dall'allineamento degli indirizzi del bus, tratteremo dapprima la dimensionabilità dinamica del bus e il disallineamento degli operandi. Con questi concetti ben fissati nella mente, saremo

in grado di trattare più precisamente le operazioni di lettura e di scrittura. Il processore utilizza la dimensionabilità dinamica del bus per permettere il trasferimento di un numero qualsiasi di byte (1, 2, 3 o 4) fra due o più periferiche anch'esse in grado di gestire dati di lunghezza variabile (1, 2, 3 o 4 byte). La dimensionabilità dinamica del bus gioca un ruolo chiave per quanto riguarda la possibilità di gestire il disallineamento degli operandi; mentre i processori precedenti della famiglia 68000 richiedevano che gli operandi di tipo word e di tipo long word risiedessero solo ad indirizzi pari, il 68020 permette l'accesso a operandi di qualsiasi dimensione a qualsiasi indirizzo, sia esso pari o dispari.
È da notare il fatto che quanto detto si applica solo per il fetch degli operandi. Per raggiungere il massimo dell'efficienza, il 68020 segue la regola che l'istruzione debba trovarsi ad indirizzi pari: il tentativo di accedere a un'istruzione a un indirizzo dispari provoca l'inizio di un processo di exception tramite il vettore di gestione degli errori.

LINEE DI CONTROLLO APPLICABILI

Esistono tre gruppi di linee di bus critiche per il modo in cui il processore trasferisce i dati sul bus. Queste linee sono quelle relative alle dimensioni dei dati trasferiti, (SIZ0 e SIZ1), quelle relative alla ricezione della dimensione dei dati e dell'avvenuto trasferimento ($\overline{\text{DSACK0}}$ e $\overline{\text{DSACK1}}$) e quelle relative agli indirizzi meno significativi (A0 e A1). SIZ0 e SIZ1 indicano il numero di byte che il processore vuole trasferire. La Tabella 6.1 mostra come SIZ0 e SIZ1 trasformano i valori in essi contenuti nel numero di byte coinvolto nel trasferimento.
$\overline{\text{DSACK0}}$ e $\overline{\text{DSACK1}}$ indicano l'"ampiezza" della porta che dovrà raccogliere i dati. All'inizio del ciclo di trasferimento, dopo che il processore ha espresso il desiderio di effettuare un trasferimento, la periferica associata all'indirizzo specificato deve comunicare al processore quanti byte alla volta è in grado di trattare. La Tabella 6.2 mostra come $\overline{\text{DSACK0}}$ e $\overline{\text{DSACK1}}$ trasformano i valori in essi contenuti nella dimensione della porta coinvolta nel trasferimento.

Tabella 6.1 Codifica di SIZ0/SIZ1.

SIZ0	SIZ1	Numero di byte
0	1	1
1	0	2
1	1	3
0	0	4

Tabella 6.2 Codifica di $\overline{DSACK0}/\overline{DSACK1}$.

$\overline{DSACK0}$	$\overline{DSACK1}$	Significato
H	H	Porta non pronta
H	L	Porta a 8 bit pronta
L	H	Porta a 16 bit pronta
L	L	Porta a 32 bit pronta

Tabella 6.3 Codifica di A0-A1.

A0	A1	Offset rispetto all'indirizzo corretto di una long word
0	0	+0 byte (word e long word)
0	1	+1 byte
1	0	+2 byte (word)
1	1	+3 byte

A1 e A0 indicano l'allineamento del trasferimento dati. A seconda dello stato, il trasferimento può avvenire a indirizzi longword-allineati, a indirizzi pari o a indirizzi indifferentemente pari o dispari. La Tabella 6.3 mostra i possibili stati di A0 e A1.
A seconda dell'allineamento e del numero di byte che devono essere trasferiti, il processore può usare da 1 a 4 cicli di trasferimento: i trasferimenti possono essere effettuati in gruppi di 1, 2, 3 o 4 byte. A seconda dell'ampiezza della porta, il processore può usare tutto o parte del bus dati per effettuare il trasferimento. Ad esempio per trasferire una long word a un indirizzo dispari potrebbe essere necessario un trasferimento di tipo 1-byte/2-byte/1-byte oppure uno di tipo 1-byte/3-byte a seconda dell'indirizzo.
Per comprendere la complessità del controllo dimensione/allineamento, è più conveniente confrontare il numero di byte che restano da trasferire (come indicato da SIZ0/SIZ1) con i diversi allineamenti e con le diverse dimensioni della porta (come indicato da A0/A1 e $\overline{DSACK0}/\overline{DSACK1}$). Per diversificare i diversi byte durante un trasferimento useremo la notazione indicata in Figura 6.8.

MULTIPLEXAGGIO DEL BUS DATI

Per facilitare il trasferimento dei dati in ognuna delle condizioni descritte, il processore fa uso di un multiplexer per il bus dati. A seconda della dimensione dei dati da trasferire e dell'allineamento dell'indirizzo, il multiplexer usa parti differenti del bus dati per ogni byte del trasferimento. La Tabella

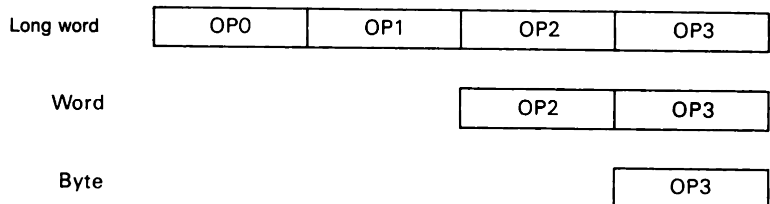


Figura 6.8 Convenzioni per etichettare i byte.

6.4 riassume le posizioni dei dati sul bus dati per ciascuna delle quattro possibili dimensioni del trasferimento.
È importante notare che tutte le porzioni del bus dati contengono dei dati, come mostrato in Tabella 6.5. In ogni caso la logica esterna dovrebbe possedere i mezzi per ignorare questi dati extra.
Un trasferimento di dati sul bus avviene come un processo suddivisibile in tre passi:

1. Il processore attiva l'informazione circa l'indirizzo e le dimensioni: inizialmente esso assume che la periferica possa accettare o trasferire tut-

Tabella 6.4 Attività del bus dati in caso di porte da 8, 16 e 32 bit.

Dimensione del trasferimento	A1	A0	Parte attiva del bus dati			
			D31-D24	D23-D16	D15-D8	D7-D0
1 byte (SIZ1/SIZ0 = 01)	0	0	BWL	—	—	—
	0	1	B	WL	—	—
	1	0	BW	—	L	—
	1	1	B	W	—	L
2 byte (SIZ1/SIZ0 = 10)	0	0	BWL	WL	—	—
	0	1	B	WL	L	—
	1	0	BW	W	L	L
	1	1	B	W	—	L
3 byte (SIZ1/SIZ0 = 11)	0	0	BWL	WL	L	—
	0	1	B	WL	L	L
	1	0	BW	W	L	L
	1	1	B	W	—	L
4 byte SIZ1/SIZ0 = 00)	0	0	BWL	WL	L	L
	0	1	B	WL	L	L
	1	0	BW	W	L	L
	1	1	B	W	—	L

(B = porta di 8 bit, W = porta di 16 bit, L = porta di 32 bit)

Tabella 6.5 Multiplexer dal bus dati interno a quello esterno.

			Dato attivato sul bus dati			
Dimensione del trasferimento	**A1**	**A0**	**D31-D24**	**D23-D16**	**D15-D8**	**D7-D0**
1 byte (SIZ1/SIZ0 = 01)	X	X	OP3	OP3	OP3	OP3
2 byte (SIZ1/SIZ0 = 10)	X	0	OP2	OP3	OP2	OP3
	X	1	OP2	OP2	OP3	OP2
3 byte (SIZ1/SIZ0 = 11)	0	0	OP1	OP2	OP3	OP1
	0	1	OP1	OP1	OP2	OP3
	1	0	OP1	OP2	OP1	OP2
	1	1	OP1	OP1	OP1	OP1
4 byte (SIZ1/SIZ0 = 00)	0	0	OP0	OP1	OP2	OP3
	0	1	OP0	OP0	OP1	OP2
	1	0	OP0	OP1	OP0	OP1
	1	1	OP0	OP0	OP1	OP0

(X = ininfluente)

ti i dati all'interno di un ciclo. In caso di scrittura, esso carica il bus dati come mostrato in Tabella 6.4. In caso di lettura esso effettua una wait sulla periferica per comunicarle dove cercare il dato sul bus dati.

2. La periferica accetta o carica tutti i dati che può e setta $\overline{DSACK0}/\overline{DSACK1}$ di conseguenza.

3. Il processore sottrae da SIZ0 e SIZ1 e aggiunge ad A0 e A1 il numero dei byte trasferiti. Se rimangono altri byte il processore esegue i passi da 1 a 3 fino a che non sono stati trasferiti tutti i byte.

ESEMPI DI ALLINEAMENTO/DIMENSIONAMENTO

Ora mostreremo parecchi esempi di trasferimento dati. Il primo caso è quello del trasferimento di una long word usando una periferica a 32 bit a un indirizzo longword-allineato (A1/A0=00). Il trasferimento dei dati richiede un ciclo e i segnali sono come segue:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	0	0	0	0	OP0	OP1	OP2	OP3

Il secondo esempio è il trasferimento di una long word con una periferica di 32 bit a un indirizzo dispari (A1/A0 = 01). Il trasferimento dati richiede sue cicli come segue:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	0	1	0	0	–	OP0	OP1	OP2
2	0	0	0	1	OP3	–	–	–

Il prossimo esempio è quello del trasferimento di un'altra long word verso una periferica di 32 bit a un indirizzo dispari (A1/A0=11). Il trasferimento dati richiede ancora due cicli:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	1	1	0	0	–	–	–	OP0
2	0	0	1	1	OP1	OP2	OP3	–

Il quarto esempio è quello del trasferimento di una long word verso una periferica a 16 bit a un indirizzo pari (A1/A0=10). Il trasferimento richiede due cicli:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	1	0	0	0	OP0	OP1	–	–
2	0	0	1	0	OP2	OP3	–	–

L'esempio seguente è il trasferimento di una long word con una periferica a 8 bit a un indirizzo longword-allineato (A1/A0=00). Il trasferimento dati richiede quattro cicli:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	0	0	0	0	OP0	–	–	–
2	0	1	1	1	OP1	–	–	–
3	1	0	1	0	OP2	–	–	–
4	1	1	0	1	OP3	–	–	–

Il sesto esempio è il trasferimento di una word verso una periferica di 16 bit a un indirizzo dispari (A1/A0=01). Il trasferimento dati richiede due cicli:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	0	1	1	0	–	OP2	–	–
2	1	0	0	1	OP3	–	–	–

L'esempio finale è il trasferimento di una word con una periferica di 32 bit a un indirizzo dispari (A1/A0=01). Il trasferimento dati richiede un ciclo come illustrato nel seguito:

Trasferimento	A1	A0	SIZ1	SIZ0	D31-D24	D23-D16	D15-D8	D7-D0
1	0	1	1	0	–	OP2	OP3	–

Come potete vedere dagli esempi, i trasferimenti dei dati sono piuttosto semplici quando vengono esaminati dal punto di vista della Tabella 6.4. Dovrebbe essere ovvio che i trasferimenti più efficienti sono quelli che coinvolgono periferiche a 32 bit. Inoltre i trasferimenti di long word sono trattati in modo molto più efficiente quando l'indirizzo è longword-allineato (A1/A0=00).

6.3 Temporizzazione del ciclo di lettura

Facendo tesoro delle informazioni ricevute, possiamo esaminare in dettaglio la temporizzazione del ciclo di lettura.

LETTURA IN UN CICLO DI UNA LONG WORD

Per leggere una long word da una porta di 32 bit, il processore inizia la lettura con i seguenti segnali: setta R/$\overline{W}$ per la lettura, emette i codici di funzione appropriati ed emette gli indirizzi desiderati sul bus indirizzi di output settando di conseguenza i segnali di dimensionamento. Il processore attiva quindi i segnali di partenza del ciclo ($\overline{OCS}$ ed $\overline{ECS}$), lo strobe indirizzi ($\overline{AS}$), lo strobe dati ($\overline{DS}$) e la linea di abilitazione del buffer dati ($\overline{DBEN}$).
Quando la periferica "sente" gli strobe dati e indirizzi, essa decodifica l'indirizzo sul bus indirizzi, pone il dato appropriato sul bus dati e attiva i segnali di trasferimento dei dati e il segnale ricezione delle dimensioni ($\overline{DSACK0}$/$\overline{DSACK1}$), stante a indicare che tutti e quattro i byte sono sul bus dati.
Se il processore non vede $\overline{DSACK0}$/$\overline{DSACK1}$ entro la fine dello stato 2, esso inserisce automaticamente degli stati di wait nel ciclo. Quando il processore "sente" i segnali $\overline{DSACK0}$/$\overline{DSACK1}$, esso cattura il dato dal bus dati e nega $\overline{DS}$, $\overline{AS}$ e $\overline{DBEN}$. A questo punto la periferica dovrebbe fermarsi cedendo il dato e negando sia $\overline{DSACK0}$ che $\overline{DSACK1}$: ora il processore può dare inizio al ciclo successivo. La Figura 6.9 illustra la temporizzazione per questa operazione.

LETTURA DI BYTE E WORD

Le letture di byte e word da una porta di 32 bit funziona in modo pressoché analogo al caso precedente: essa differisce però nello stato delle sue linee degli indirizzi meno significativi (A0 e A1) e dei segnali di dimensionamento (SIZ0 e SIZ1). La Figura 6.10 illustra sia la lettura di una word che quella di un byte da un indirizzo longword-allineato di una porta di 32 bit.

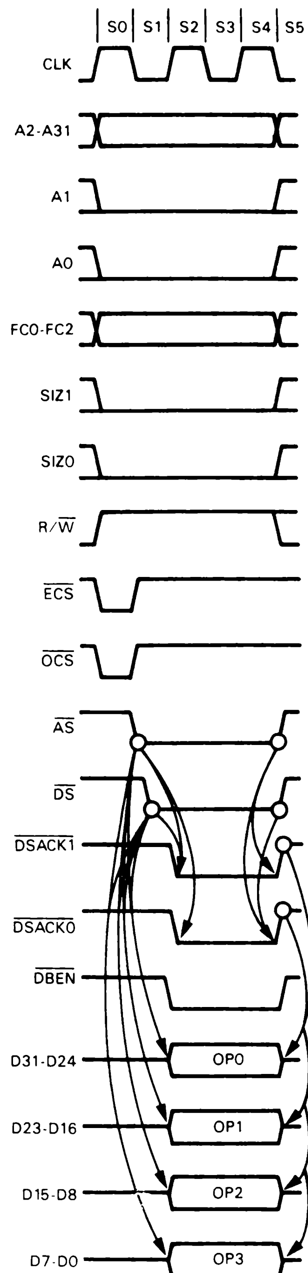


Figura 6.9 Temporizzazione della lettura di una word (porta di 32 bit).

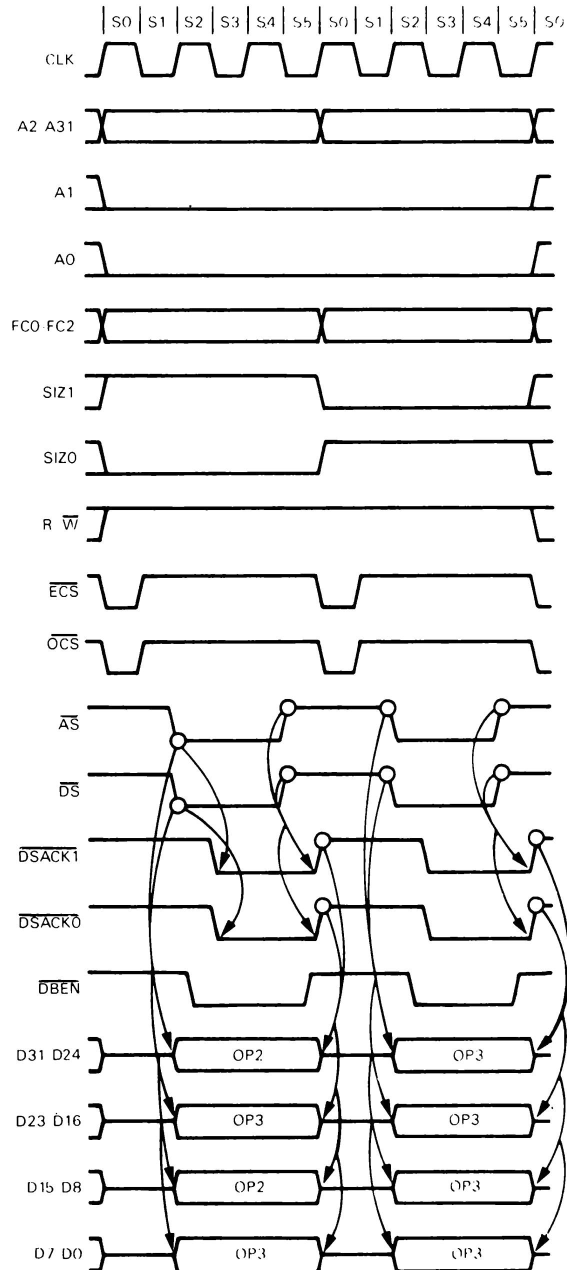


Figura 6.10 Temporizzazione della lettura di una word e di un byte (porta di 32 bit).

LETTURE DISALLINEATE

Come indicato in precedenza, l'accesso ai dati che non si trovano a indirizzi longword-allineati comporta il multiplexaggio del bus dati e il possibile uso di cicli multipli. Il processore carica e si aspetta i dati suddivisi sul bus dati come mostrato in Tabella 6.4. L'accesso tramite più cicli funziona in modo simile all'accesso tramite ciclo singolo, ma con le seguenti differenze: (1) il processore attiva il segnale $\overline{OCS}$ solo una volta, ma attiva il segnale $\overline{ECS}$ all'inizio di ogni fetch; (2) SIZ0/SIZ1 e A1/A0 cambiano ad ogni accesso per indicare la dimensione e l'indirizzo della rimanente parte dell'operando.
La Figura 6.11 mostra l'operazione di temporizzazione per la lettura di una word da una porta di 16 bit a un indirizzo dispari. Questo trasferimento corrisponde al sesto esempio descritto poco sopra nel paragrafo "Esempi di allineamento/dimensionamento".

6.4 Temporizzazione del ciclo di scrittura

SCRITTURA IN UN CICLO DI UNA LONG WORD

Per la scrittura di una long word su una porta di 32 bit, il processore inizia la scrittura con i segnali seguenti: il processore setta R/$\overline{W}$ in modalità "scrittura" (basso), emette gli appropriati codici di funzione e l'indirizzo desiderato sul bus indirizzi settando di conseguenza il segnale di dimensionamento. Il processore attiva i segnali di partenza del ciclo ($\overline{OCS}$ ed $\overline{ECS}$) e lo strobe indirizzi ($\overline{AS}$), emette i dati sul bus dati e attiva la linea di abilitazione del buffer dati ($\overline{DBEN}$).
Quando la periferica "sente" lo strobe indirizzi comunica al processore la propria "ampiezza" attraverso i segnali $\overline{DSACK0}$/$\overline{DSACK1}$. Se la periferica non risponde, il processore inserisce automaticamente stati di wait fino a che entrambi i $\overline{DSACK}$ vengono attivati oppure fino a che la logica esterna segnala un errore di bus ($\overline{BERR}$). Quando il processore riceve il segnale $\overline{DSACK0}$/$\overline{DSACK1}$, esso attiva lo strobe dati ($\overline{DS}$) e la periferica deve catturare il dato.
Il processore quindi nega $\overline{DS}$, $\overline{AS}$ e $\overline{DBEN}$ e rimuove il dato dal bus dati. Quando la periferica si accorge di ciò, essa dovrebbe negare sia $\overline{DSACK0}$ che $\overline{DSACK1}$. Il processore può quindi eseguire il ciclo successivo. La Figura 6.12 mostra la temporizzazione per la scrittura di una long word.

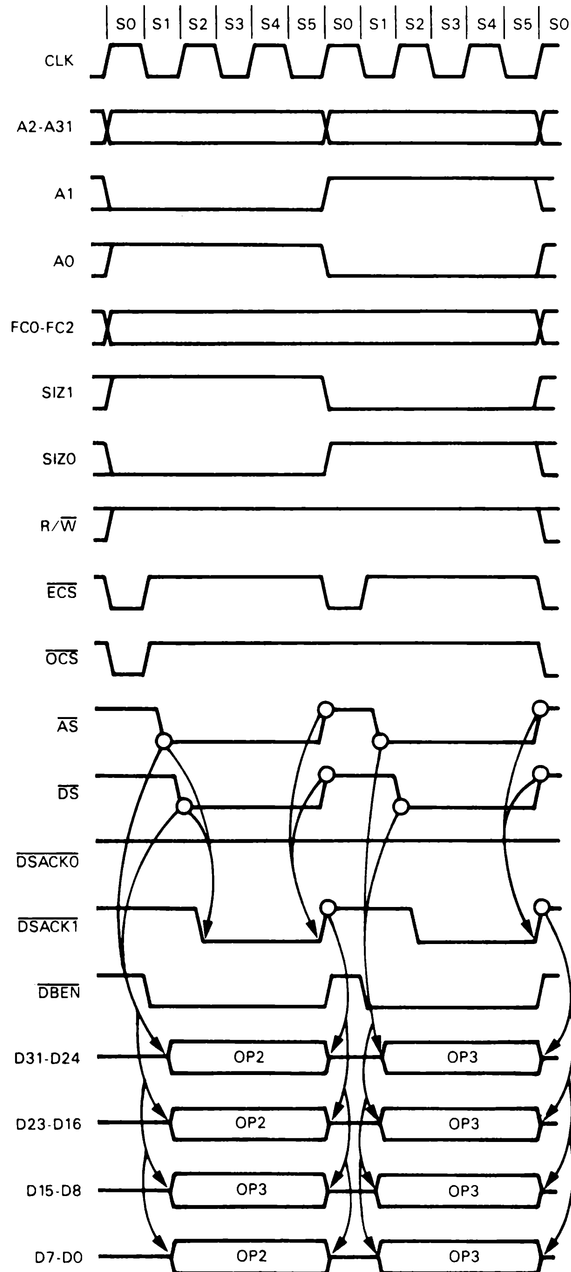


Figura 6.11 Temporizzazione della lettura di una word disallineata (porta di 16 bit).

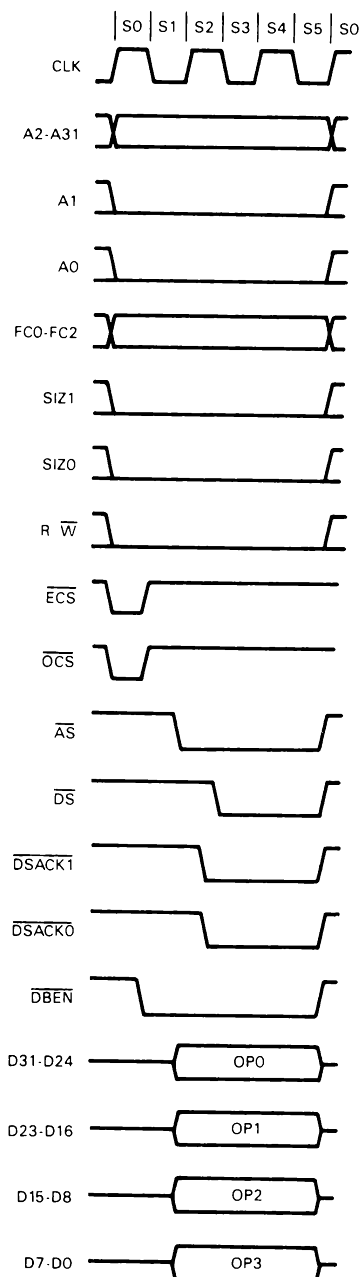


Figura 6.12 Temporizzazione della lettura di una long word (porta di 32 bit).

SCRITTURA DI BYTE E WORD

La scrittura di byte e word su una porta di 32 bit funziona in modo pressoché analogo al caso precedente: essa differisce però nello stato delle due linee degli indirizzi meno significativi (A0 e A1) e dei segnali di dimensionamento (SIZ0 e SIZ1). La Figura 6.10 illustra sia la lettura di una word che quella di un byte a indirizzi longword-allineati di una porta di 32 bit. La Figura 6.13 mostra la temporizzazione per la scrittura di un byte e di una word.

SCRITTURE DISALLINEATE

Come indicato in precedenza, l'accesso ai dati a indirizzi longword-allineati comporta il multiplexaggio del bus dati e il possibile uso di cicli multipli. Il processore carica e si aspetta i dati suddivisi sul bus dati come mostrato in Tabella 6.4. L'accesso tramite più cicli funziona in modo simile all'accesso tramite ciclo singolo, ma con le seguenti differenze: (1) il processore attiva il segnale $\overline{OCS}$ solo una volta, ma attiva il segnale $\overline{ECS}$ all'inizio di ogni fetch; (2) SIZ0/SIZ1 e A1/A0 cambiano a ogni accesso per indicare la dimensione e l'indirizzo della rimanente parte dell'operando.
La Figura 6.14 mostra l'operazione di temporizzazione per la scrittura di una word su una porta di 16 bit a un indirizzo dispari. Questo trasferimento corrisponde al sesto esempio descritto sopra nel paragrafo "Esempi di allineamento/dimensionamento".

TEMPORIZZAZIONE DEL CICLO LETTURA-MODIFICA-SCRITTURA

Come nel caso dei membri precedenti della famiglia 68000, il 68020 ha un ciclo indivisibile di lettura-modifica-scrittura. Lo scopo di questa classe di cicli è quello di garantire che un operando non venga inaspettatamente modificato nel periodo di tempo che intercorre fra l'inizio e la fine del ciclo. Questa precauzione è particolarmente utile in sistemi multiprocessore in cui un potenziale master del bus potrebbe interrompere il processore che ne ha attualmente il controllo (attraverso una richiesta del bus $\overline{BR}$) mentre quest'ultimo sta testando e modificando l'operando.
I precedenti processori 68000 possedevano solo un'istruzione che usava il ciclo lettura-modifica-scrittura, ovvero l'istruzione di Test and Set (TAS). Il 68020 aggiunge un secondo tipo di istruzioni, ovvero quelle di confronto e scambio (CAS e CAS2).
Notate che mentre l'istruzione TAS funziona su un operando delle dimensioni di un byte, le istruzioni CAS possono operare su operandi di qualsiasi dimensione. Poiché le istruzioni CAS possono operare su fino a quattro byte a seconda dell'indirizzo e delle dimensioni delle porte, il processore può ri-

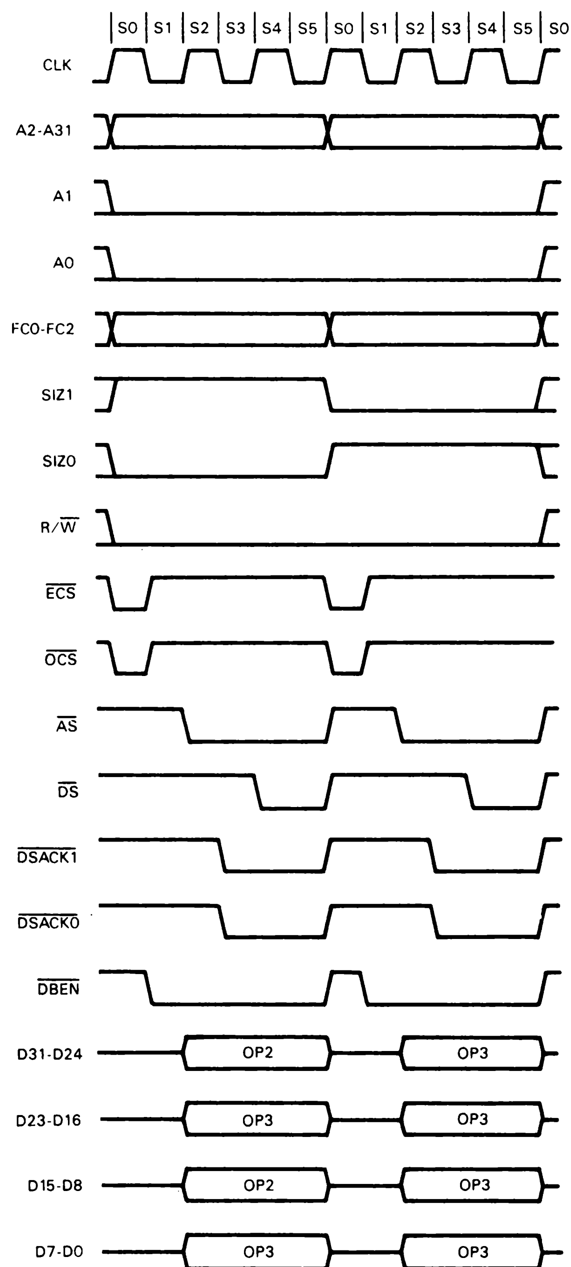


Figura 6.13 Temporizzazione della lettura di una word e di un byte (porta di 32 bit).

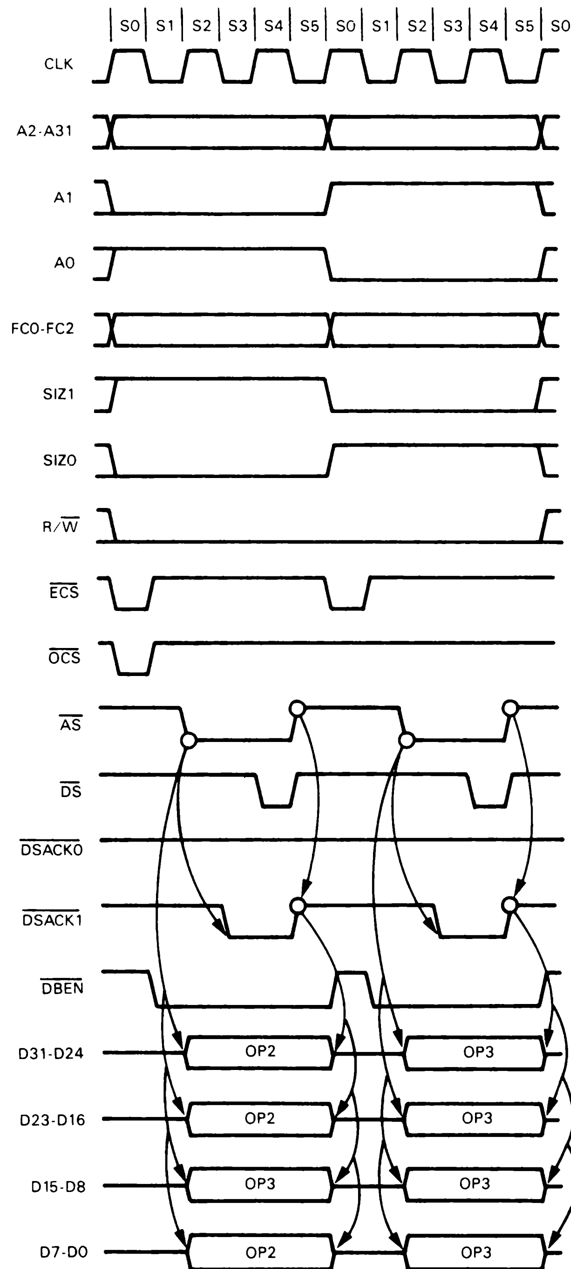


Figura 6.14 Temporizzazione della lettura di una word disallineata (porta di16 bit).

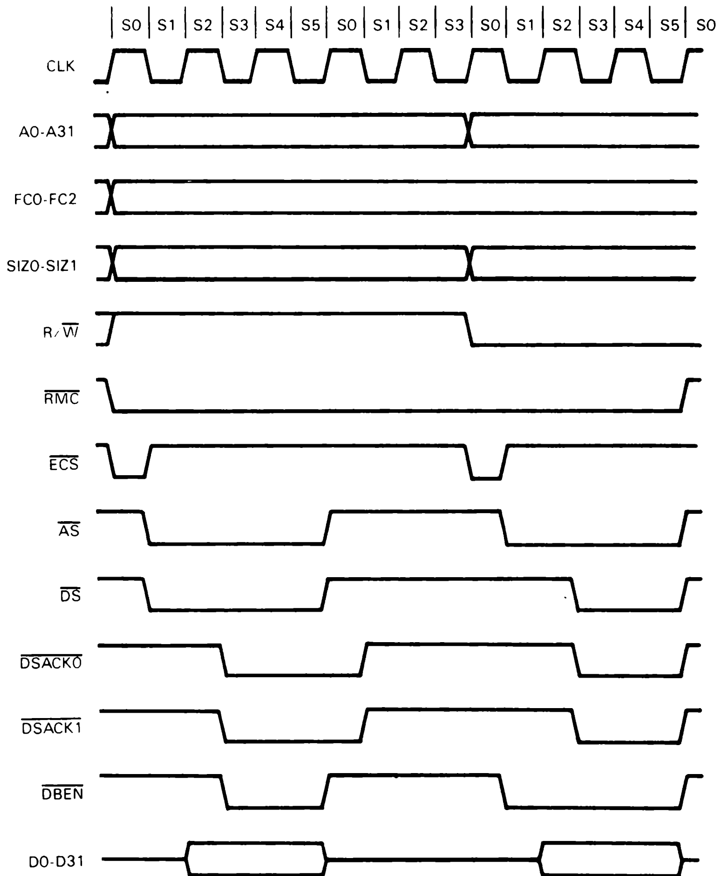


Figura 6.15 Temporizzazione di lettura-modifica-scrittura (istruzione CAS, porta di 32 bit).

chiedere molti cicli di bus per completare l'operazione. Dal momento che il bus è l'unica proprietà del processore durante l'istruzione, ciò ha un serio impatto sui meccanismi di arbitraggio del bus. Idealmente tutti i semafori cui accedono CAS o CAS2 dovrebbero trovarsi a indirizzi longword-allineati. Questa restrizione non si applica all'istruzione TAS in quanto essa accede solo a un byte di dati.

La Figura 6.15 mostra la temporizzazione per una istruzione CAS su una porta di 32 bit.

6.5 Caratteristiche comuni della famiglia 68000

Finora abbiamo considerato la temporizzazione delle operazioni di lettura/scrittura separando il caso del 68020 da quello degli altri membri della famiglia 68000. Il resto del presente capitolo tratterà le operazioni sul bus comuni a tutti i processori della famiglia 68000.

6.6 Operazioni di reset

RESET HARDWARE

I processori 68000 hanno un input di reset asincrono. Sul 68000, 68008, 68010 e 68012 attivando i segnali $\overline{\text{HALT}}$ e $\overline{\text{RESET}}$ per almeno 100 millisecondi, il processore viene forzato a resettarsi. Il 68020 funziona in modo analogo, con la differenza che il segnale di $\overline{\text{RESET}}$ deve essere attivato da solo. Dopo aver negato questi segnali, il processore esegue le operazioni seguenti:

1. Il 68000 carica la prima long word (offset 0) della tabella dei vettori nello SSP (System Stack Pointer o stack pointer di sistema). Esso carica inoltre la seconda long word (offset 4) nel program counter (PC).
2. Il 68000 setta la maschera di interrupt nel registro di stato (SR) a tutti uno. Ciò significa che solo i segnali di interrupt non mascherabili possono interrompere il processore; il bit supervisore del registro di stato viene settato anche per indicare che il processore opererà in modalità supervisore: oltre a ciò i bit di trace vengono puliti. Sul 68010, 68012 e 68020 il VBR (Vector Base Register o registro base dei vettori) viene posto all'indirizzo 0000 0000. Sul 68020 viene disabilitata la cache istruzioni e viene pulito il bit master del registro di stato.
3. Il 68000 inizia l'esecuzione del programma all'indirizzo indicato dal nuovo contenuto del program counter.

Notate che i valori contenuti nei registri general-purpose dati e indirizzi restano indefiniti a seguito di un reset di sistema.

RESET SOFTWARE

Ricorderete che il segnale $\overline{\text{RESET}}$ è bidirezionale; quando il 68000 esegue un'istruzione di $\overline{\text{RESET}}$, esso attiva la linea $\overline{\text{RESET}}$ per 124 cicli di clock (512 nel 68020). Questa istruzione non ha alcun effetto sullo stato interno

del processore e nessuno dei suoi registri viene alterato. In questo reset software il processore invia un comando di reset a tutte le periferiche collegate alla linea $\overline{\text{RESET}}$.

6.7 Lo stato di halt

HALT INIZIATO ESTERNAMENTE

Il 68000 può essere forzato in uno stato di halt attivando dall'esterno la linea di $\overline{\text{HALT}}$. Dopo il completamento del ciclo di bus corrente, il bus indirizzi, il bus dati e i codici funzione vanno in uno stato ad alta impedenza e il processore ferma l'esecuzione fino a che non si accorge della negazione del segnale di halt. Notate che il processore non possiede alcun meccanismo per comunicare alla logica esterna la ricezione dello stato di halt.
Sebbene nello stato di halt il processore non stia eseguendo alcuna istruzione, funzionano ancora i circuiti dedicati all'arbitraggio del bus. In ogni caso, dal momento che il 68000 ha rilasciato il bus, viene soddisfatta immediatamente ogni richiesta di acquizione del bus proveniente dalla logica esterna. Tratteremo l'arbitraggio del bus più avanti nel capitolo.

PASSO SINGOLO CON $\overline{\text{HALT}}$

L'esecuzione della maggior parte delle istruzioni richiede più cicli di bus per rintracciare l'istruzione e gli operandi. Dal momento che il processore risponderà al segnale di $\overline{\text{HALT}}$ dopo aver completato un qualsiasi ciclo di bus, la sequenza di halt può occorrere fra due istruzioni o nel mezzo di una istruzione singola. È perciò possibile usare l'input del segnale di $\overline{\text{HALT}}$ per implementare una modalità di esecuzione a passo singolo. Dal momento che il processore non fornisce alcuna indicazione circa il suo stato, questa modalità è particolarmente utile nel caso in cui si voglia effettuare il debugging dell'hardware. Ricordate che la modalità di esecuzione a passo singolo è disponibile tramite la funzione di trace attraverso il bit di trace del registro di stato. Descriveremo il tracing attivato via software nel prossimo capitolo.

OUTPUT DI HALT

Come ricorderete il segnale di $\overline{\text{HALT}}$ è bidirezionale. Se il 68000 incontra un doppio bus fault (un errore che avviene sul bus cercando di effettuare il fetch di un vettore di exception relativo all'errore di bus), il processore

entra nello stato di halt e attiva il segnale di $\overline{HALT}$ per indicare un catastrofico fallimento. L'unico modo per far ripartire il processore è quello di effettuare un'operazione di reset hardware.

6.8 Lo stato di stop

Lo stato di stop è simile allo stato di halt in quanto il processore essenzialmente non fa niente. Dopo aver eseguito lo STOP (un'istruzione privilegiata), il processore carica in modalità a indirizzamento immediato nel registro di stato il dato che si trova subito dopo l'istruzione di stop; esso carica quindi nel program counter l'indirizzo dell'istruzione successiva ed entra nello stato di stop.
A differenza dello stato di halt, il processore non emette alcun particolare segnale per identificare il proprio stato. Inoltre, mentre lo stato di halt indica la presenza di un errore disastroso che può essere risolto solo resettando il processore, lo stato di stop può essere lasciato attraverso un processo di exception (corrispondente alla generazione di un interrupt).

6.9 Riesecuzione del ciclo di bus

Come menzionato in precedenza, il 68000 può rispondere in due modi a un errore sul bus di sistema (riconosciuto grazie all'attivazione di $\overline{BERR}$). Esso può eseguire un processo di exception (che tratteremo nel Capitolo 7), o può tentare di rieseguire il ciclo di bus che ha causato l'errore. Se $\overline{BERR}$ viene attivato da solo, il processore dà inizio al processo di exception; se il segnale $\overline{BERR}$ è accompagnato dal segnale di $\overline{HALT}$, il 68000 riconosce la richiesta di rieseguire il ciclo di bus.
Il 68000 provvede a completare il ciclo che era in esecuzione e poi entra nello stato di halt. Il bus indirizzi, il bus dati e i codici di funzione sono posti in uno stato ad alta impedenza e il processore rimane in stato di halt fino a che verranno negati sia $\overline{HALT}$ che $\overline{BERR}$. Notate che $\overline{BERR}$ dovrebbe venire negato almeno un ciclo di clock prima che venga negato $\overline{HALT}$, in modo da impedire che il 68000 interpreti il segnale $\overline{BERR}$ come la segnalazione di un ulteriore errore sul bus che, in quest'ultimo caso, sarebbe da gestire via software.
Dopo la negazione di $\overline{HALT}$, il 68000 provvederà a ripetere il ciclo che era in esecuzione al momento del ricevimento della richiesta di riesecuzione: ovviamente verranno eseguite esattamente le stesse operazioni sugli stessi dati.

RIESECUZIONE CONCLUSA FAVOREVOLMENTE

Se l'operazione di riesecuzione del ciclo di bus si conclude positivamente, il segnale DTACK verrà attivato entro un certo periodo di tempo. In caso contrario, ovvero di fallimento dell'operazione di riesecuzione, la logica esterna può continuare a chiedere la riesecuzione del ciclo per un numero infinito di volte usando la combinazione di BERR e HALT. È da notare il fatto che, se state usando il metodo di gestione delle eccezioni via software (attivando BERR da solo), due successivi errori sul bus (un "doppio bus fault") daranno luogo a un errore catastrofico; il 68000 entrerà automaticamente nello stato di halt e vi rimarrà fino a che non verrà resettato.
Se nel momento in cui incontra un bus fault il 68000 sta eseguendo un ciclo di lettura-modifica-scrittura, esso non rieseguirà il ciclo in quanto il ciclo lettura-modifica-scrittura viene eseguito solo durante l'istruzione di Test and Set (TAS). La natura di questa istruzione richiede una completa integrità del ciclo in cui viene eseguita, integrità che potrebbe essere violata se i cicli di bus venissero rieseguiti. Se la logica esterna richiedesse la riesecuzione del ciclo lettura-modifica-scrittura, il 68000 inizierebbe automaticamente la routine di exception relativa all'errore sul bus.

6.10 Logica di arbitraggio del bus

La logica di arbitraggio del bus implementata sul 68000 è piuttosto semplice. Esso non possiede alcuna tabella di priorità per quanto riguarda l'accesso al bus da parte dei dispositivi esterni: il processore suppone infatti di essere il dispositivo con la minore priorità all'interno del sistema in quanto è sempre disposto a cedere il bus a qualsiasi dispositivo ne faccia richiesta (previa ovviamente conclusione delle operazioni sul bus medesimo).
Il 68000 quindi permette agli altri dispositivi di usare il bus fra le varie istruzioni e fra i cicli di bus delle varie istruzioni. Dal momento che non vi è alcuna modalità di arbitraggio predefinita, dovrebbe esistere una qualche logica di arbitraggio esterna che permetta, in sistemi di complessità qualsiasi, la gestione di una coda di priorità per evitare che un dispositivo ad alta priorità venga preceduto nell'accesso al bus da un dispositivo a priorità inferiore.
Esistono tre segnali associati alla logica di arbitraggio del bus; essi sono il segnale di richiesta del bus (BR), il segnale di cessione del bus (BG) e il segnale indicante il fatto che il bus è stato acquisito (BGACK). Quando il 68000 sta usando il bus di sistema senza alcun dispositivo che ne richieda l'accesso, i segnali di input BR e BGACK sono inattivi, mentre l'output del segnale BG è negato.

TEMPORIZZAZIONE DELL'ARBITRAGGIO

La Figura 6.16 illustra la temporizzazione dell'arbitraggio del bus effettuata dal 68000. L'arbitraggio comincia quando un dispositivo esterno attiva $\overline{BR}$: a questo punto esso risponderà al segnale attivando $\overline{BG}$ un ciclo di clock più tardi. L'unica eccezione a questa risposta immediata si ha quando il 68000 si trova nella parte iniziale del ciclo di bus ma non ha ancora attivato $\overline{AS}$. In questo caso il 68000 aspetta un periodo di clock dopo l'attivazione di $\overline{AS}$ prima di attivare $\overline{BG}$. Il tempo di risposta in questo caso sarà al massimo pari a tre periodi di clock.

DETERMINAZIONE DELLA DISPONIBILITÀ DEL BUS

Il segnale di cessione del bus attivato da solo non indica che il bus è immediatamente disponibile per il dispositivo che ha effettuato la richiesta, in

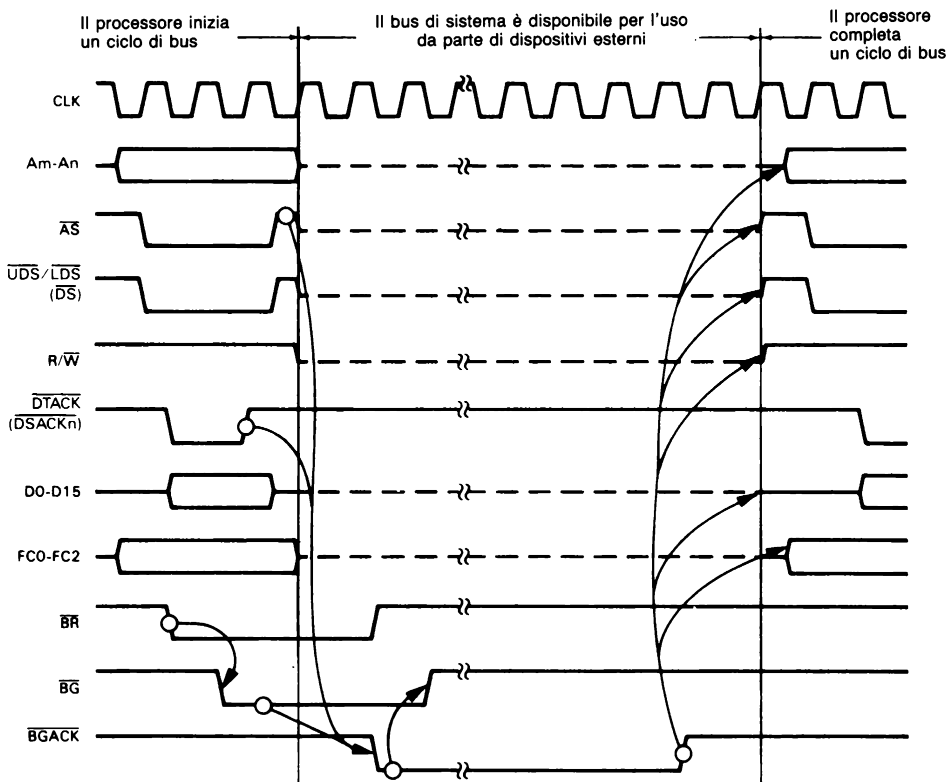


Figura 6.16 Temporizzazione dell'arbitraggio del bus.

quanto il 68000 potrebbe avere ancora bisogno del bus per completare il ciclo di bus corrente. Per questo motivo il dispositivo che ha effettuato la richiesta deve esaminare parecchi altri segnali per determinare in quale momento gli verrà ceduto effettivamente il bus.
In primo luogo il dispositivo esterno deve aspettare fino al momento in cui verrà negato $\overline{AS}$: ciò starà a significare che il 68000 ha completato il ciclo di bus corrente. Il dispositivo che ha richiesto il bus deve aspettare che il segnale $\overline{DTACK}$ (o la coppia $\overline{DSACK0}$/$\overline{DSACK1}$ per il 68020) venga negato, in quanto ciò indica che il dispositivo coinvolto nel ciclo di bus corrente non sta più usando il bus. (In alcuni sistemi potrebbe non essere necessario controllare il segnale $\overline{DTACK}$ o la coppia $\overline{DSACK0}$/$\overline{DSACK1}$. Un caso potrebbe essere quello in cui la temporizzazione del sistema è tale da assicurare sempre che tutti i dispositivi esterni siano off quando il segnale $\overline{AS}$ è negato.)
Infine, il dispositivo richiedente deve controllare lo stato del segnale $\overline{BGACK}$: se questo segnale viene attivato, esso indica che il bus di sistema è già stato ceduto ad altri dispositivi nel sistema e che questi ultimi non hanno ancora finito di lavorare con esso. Contrariamente se $\overline{BGACK}$ è settato a false, il bus di sistema sarà disponibile alla fine del ciclo corrente.
Dopo che tutti i segnali sono stati ricevuti correttamente, il dispositivo richiedente deve attivare $\overline{BGACK}$; ciò informa il processore che il dispositivo richiedente ha preso possesso del bus. Noterete in Figura 6.16 che il 68000 non aspetta il segnale $\overline{BGACK}$ prima di cedere il controllo del bus: il bus dati e quello indirizzi, i codici di funzione, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$ (o $\overline{DS}$ per il 68008 e 68020) e R/$\overline{W}$ sono posti in uno stato ad alta impedenza non appena il processore ha completato il ciclo di bus che stava eseguendo.
Il dispositivo che ha ottenuto il controllo del bus deve lasciare attivo $\overline{BGACK}$ fino a quando ha intenzione di tenere il bus. Durante il periodo di tempo in cui un dispositivo esterno mantiene il controllo del bus, la logica esterna dovrebbe impedire dei conflitti sul bus tenendo sotto controllo $\overline{BGACK}$; a questo punto il comportamento di $\overline{BR}$ e $\overline{BG}$ è ininfluente. In ogni caso comunque, prima di negare $\overline{BGACK}$, il dispositivo dovrebbe negare il suo $\overline{BR}$ per evitare richieste di bus non corrette.

Riottenimento del bus

Il 68000 manterrà la proprie linee in uno stato ad alta impedenza fino a che il segnale $\overline{BGACK}$ verrà negato. A questo punto il processore è libero di iniziare un altro ciclo di bus. Notate che se un'altra richiesta di bus è in attesa di essere evasa, il 68000 la soddisferà immediatamente, senza compiere nel frattempo alcun ciclo di bus.

7

Gestione delle exception

Il processo di exception è quel particolare aspetto del processo di esecuzione che avviene in occasione di particolari eventi all'interno del microprocessore. Questi eventi speciali possono essere errori di indirizzamento, bus fault, tentativi di esecuzione di istruzioni privilegiate mentre ci si trova in modalità utente e tentativi di effettuare divisioni per zero. Le exception però non vengono generate solo da condizioni di errore: difatti anche gli interrupt da parte delle periferiche, reset hardware e interrupt programmati danno luogo a exception.

7.1 Modalità operative

Prima di procedere alla descrizione del sistema di gestione delle exception è necessario trattare le modalità operative del 68000, dal momento che queste ultime influenzano il meccanismo di gestione delle exception. Come già menzionato in precedenza, il 68000 può operare in modalità supervisore o in modalità utente. Quando il 68000 viene resettato tramite $\overline{\text{RESET}}$, esso inizia a lavorare in modalità supervisore. Il processore resta in questa modalità fino a che non viene eseguita una delle istruzioni seguenti:

- RTE ritorno dalla exception
- MOVE copia nel registro di stato
- ANDI AND immediato con registro di stato
- EORI OR esclusivo immediato con registro di stato

Nessuna di queste istruzioni provoca automaticamente la transizione in modalità utente: esse sono invece in grado di modificare lo stato del bit S del registro di stato che, se resettato, forza il 68000 a lavorare in modalità utente. Quando il 68000 sta lavorando in modalità utente, l'unico modo per farlo ritornare in modalità supervisore è una exception. Quando incomincia il processo di exception, il processore rientra in modalità supervisore. Quando il processore ha completato il processo di exception esso esegue un'istruzione di ritorno da exception (RTE) per ritornare in modalità utente.

7.2 Tipi di exception

Le exception possono essere generate in molti modi; è comunque possibile dividerle in due grandi categorie, ovvero exception generate internamente ed exception generate esternamente. Le exception generate internamente sono generate dall'esecuzione di certe istruzioni o da errori interni; le exception generate esternamente comprendono gli errori sul bus, i reset e le richieste di interrupt.

EXCEPTION GENERATE INTERNAMENTE

Possiamo dividere le exception generate internamente in tre ulteriori categorie: errori interni, trappole e funzione trace. Nel seguito troveremo un elenco di errori che provocheranno l'esecuzione di exception:

- **Errori di indirizzamento** Tutte le istruzioni devono trovarsi a indirizzi vari. Oltre a ciò, su tutti i processori tranne il 68020, gli operandi di tipo word e long word devono trovarsi a indirizzi pari. Un tentativo di accedere a operandi o istruzioni mal allineate provocherà l'attivazione del processo di exception.

- **Violazioni di privilegi** Come menzionato in precedenza certe istruzioni sono riservate a routine in modalità supervisore. Queste istruzioni permettono l'accesso a risorse che potrebbero compromettere l'esecuzione se usate da un programma utente. L'esecuzione da parte dell'utente di questo tipo di istruzioni potrebbe provocare un processo di exception.

- **Istruzioni illegali e non implementate** Dal momento che ogni istruzione è lunga 16 bit, esistono delle combinazioni di bit che non corrispondono ad alcuna istruzione. Il tentativo da parte del programma di eseguire una di queste combinazioni provoca l'esecuzione di un processo di exception. Due combinazioni di bit vengono definite come "non im-

plementate" anziché come "illegali"; se i bit 15-12 corrispondono alla configurazione 1010 (linea "A") o 1111 (linea "F"), il processore dà inizio a un processo di exception diverso da quello iniziato in caso di istruzione illegale. Sul 68010, 68012 e 68020 i codici istruzione da $4848 a $484F identificano un tipo speciale di istruzione chiamato "breakpoint". I breakpoint verranno trattati più avanti nel capitolo.

Le trap sono exception provocate dall'esecuzione di particolari istruzioni all'interno di un programma. Esiste un'istruzione TRAP standard che viene spesso usata dai sistemi operativi per impedire ai programmi degli utenti un accesso indiscriminato alle risorse di sistema accessibili solo in modalità supervisore. Esistono altre istruzioni (CHK, CHK2, ccTRAPcc, TRAPcc, TRAPV, DIVS e DIVU) che provocano l'esecuzione di routine di exception al verificarsi di particolari condizioni quali ad esempio overflow di operazioni aritmetiche o di divisione per zero.
Il terzo tipo di exception generata internamente si verifica quando il 68000 sta operando in modo trace. Se i bit T del registro di stato sono settati, il processo di exception potrebbe potenzialmente avere inizio dopo ogni istruzione. La funzione trace è utile durante il debugging dei programmi, dal momento che rende possibile analizzare l'esecuzione del programma arrestandosi dopo ogni istruzione o dopo l'esecuzione di istruzioni chiave.

EXCEPTION GENERATE ESTERNAMENTE

Esistono tre tipi di exception generate esternamente: errore sul bus, generato dall'attivazione da parte della logica esterna di $\overline{\text{BERR}}$; reset, generato dall'attivazione da parte della logica esterna di $\overline{\text{RESET}}$; richieste di interrupt generate dall'attivazione di una o più linee di richiesta di interrupt ($\overline{\text{IPL0}}$-$\overline{\text{IPL2}}$).

7.3 Priorità delle exception

Il processore applica differenti priorità alle varie routine di exception. Le routine a priorità più elevata sono i reset, gli errori di bus e gli errori di indirizzamento. Ognuna di queste exception provocherà la terminazione immediata dell'istruzione corrente anche all'interno di un ciclo di bus. Il gruppo di exception successivo (trace, richieste di interrupt, istruzioni illegali o non implementate, violazioni di privilegi) permettono il completamento dell'istruzione corrente prima di iniziare il processo di exception. Le richieste di interrupt comportano l'introduzione di un ulteriore livello di priorità: que-

Tabella 7.1 Priorità delle exception.

Gruppo/priorità	Caratteristiche
0.0 Reset	Abortisce tutti i processi in corso (non salva lo stato dei processi al momento dell'operazione di reset)
1.0 Errore di indirizzamento	Sospende tutti i processi salvandone lo stato al momento della sospensione
1.1 Errore sul bus	
2.0 Istruzioni BKPT, CALLM, CHK, CHK2, cp Mid, violazione del protocollo con il coprocessore, cpTRAPcc, divisione per zero, RTE, RTM, TRAP, TRAPV	Il processo di exception è parte dell'esecuzione delle istruzioni
3.0 Istruzioni illegali, linea F, linea A, violazione di privilegi, preistruzioni del coprocessore	Il processo di exception comincia prima che l'istruzione venga eseguita
4.0 Postistruzioni del coprocessore	Il processo di exception comincia quando l'istruzione corrente o il precedente processo di exception sono stati completati
4.1 Trace	
4.2 Interrupt	

st'ultimo dipende dalla combinazione delle linee di richiesta di interrupt attivate e verrà peraltro trattato più avanti.
Le exception che posseggono il livello più basso di priorità sono quelle generate dall'esecuzione di istruzioni di tipo trap. Queste istruzioni possono dare inizio a un processo di exception come parte di una normale esecuzione. Tutte le exception generate da istruzioni trap hanno uguale priorità dal momento che è impossibile che due exception di questo tipo vengano generate simultaneamente.
La Tabella 7.1 elenca i tipi di exception a seconda delle priorità relative e definisce il momento in cui viene iniziato il processo di exception.

7.4 Tabella dei vettori di exception

Il sistema di gestione di exception si basa sulla tabella di vettori che occupa 1024 byte di memoria. Sul 68000 e sul 68008 questa tabella occupa gli indirizzi di memoria da 0000 a 03FF. Sui processori successivi, la posizione di

Tabella 7.2 Assegnamenti alla tabella delle exception.

Numero del vettore	Offset	Assegnamento
0	000	Reset: stack pointer per l'interrupt iniziale
1	004	Reset: program counter iniziale
2	008	Errore sul bus
3	00C	Errore di indirizzo
4	010	Istruzione illegale
5	014	Divisione per zero
6	018	Istruzione CHK, CHK2
7	01C	Istruzione cpTRAPcc, TRAPcc, TRAPV
8	020	Violazione di privilegio
9	024	Trace
10	028	Emulatore linea A
11	02C	Emulatore linea F
12	030	Riservato
13	034	Violazione del protocollo con il coprocessore
14	038	Errore di formato
15	03C	Interrupt non inizializzato
16-23	040-05C	Riservato
24	060	Interrupt spurio
25	064	Autovettore (livello 1)
26	068	Autovettore (livello 2)
27	06C	Autovettore (livello 3)
28	070	Autovettore (livello 4)
29	074	Autovettore (livello 5)
30	078	Autovettore (livello 6)
31	07C	Autovettore (livello 7)
32-47	080-0BC	TRAP 0-15
48	0C0	FPCP Branch o set al verificarsi di una condizione disordinata
49	0C4	FPCP Risultato errato
50	0C8	FPCP Divisione per zero
51	0CC	FPCP Underflow
52	0D0	FPCP Errore nell'operando
53	0D4	FPCP Overflow
54	0D8	FPCP che segnala NAN
55	0DC	Riservato
56	0E0	Configurazione PMMU
57	0E4	Operazione illegale PMMU
58	0E8	Livello di accesso PMMU
59-63	0EC-0FC	Riservato
64-255	100-3FC	Vettori definiti dall'utente

FPCP = coprocessore floating point
PMMU = MMU paginato

questa tabella in memoria è definibile dall'utente attraverso il registro base dei vettori (VBR). Dopo un reset hardware, il VBR viene azzerato per fare in modo che la tabella incominci all'indirizzo 0000.
La tabella consiste di 256 vettori di quattro byte. Ogni vettore (tranne il primo) è un indirizzo di 32 bit che verrà caricato nel program counter come parte della sequenza relativa all'exception. I primi due vettori sono riservati come program counter e stack pointer di sistema nel caso in cui il sistema stesso venga resettato. La Tabella 7.2 elenca le definizioni della tabella dei vettori. Notate che tutti i vettori vengono indirizzati a partire dall'area dati di supervisore, con l'eccezione dei vettori di reset (elementi 0 e 1).
Come potrete notare un certo numero di elementi della tabella serve a implementare le routine di exception descritte in precedenza. Certi vettori sono disponibili solo per i membri più recenti della famiglia 68000. Nei modelli precedenti essi potevano essere definiti come "riservati". I primi 64 vettori sono predefiniti o sono riservati: ciò lascia 192 vettori disponibili per usi più generali quali ad esempio le richieste di interrupt da parte dei dispositivi esterni.

7.5 Stack frame

A seconda del tipo di processore e del tipo di exception incontrata, il processore può inserire da 0 a 44 word sullo stack supervisore durante il processo di exception. Questo blocco di dati inseriti sullo stack è chiamato "stack frame". La Figura 7.1 mostra il formato generale dello stack frame di exception. Notate nella Figura 7.1 la notazione "formato". Il 68010, 68012 e il 68020 posseggono parecchi stack frame che vengono definiti dal campo format. Il 68000 e il 68008 sono meno rigidi per quanto riguarda lo stack frame in quanto essi non contengono un campo "formato" nei dati memorizzati. La Tabella 7.3 riassume le definizioni dei formati dello stack frame.

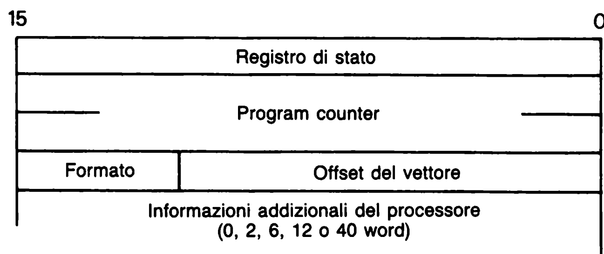


Figura 7.1 Formato generale dello stack.

Tabella 7.3 Definizione del formato degli stack frame.

Formato	Tipo
0000	Formato short (4 word)
0001	Throwaway (4 word)
0010	Exception per le istruzioni (6 word)
0011-0111	Riservato
1000	Bus fault per 68010-68012 (29 word)
1001	Medio-istruzioni del coprocessore (10 word)
1010	Short bus fault per il 68020 (16 word)
1011	Long bus fault per il 68020 (44 word)
1100-1111	Riservato

Stack frame per 68000/68008

La Figura 7.2 mostra lo stack frame creato dal 68000 e dal 68008 in occasione di una trace, una TRAP, un'istruzione illegale o non implementata, violazione di privilegi o richieste di interrupt. La Figura 7.3 mostra lo stack frame del 68000/68008 creato per una exception relativa a un errore di bus o a un errore di indirizzamento.

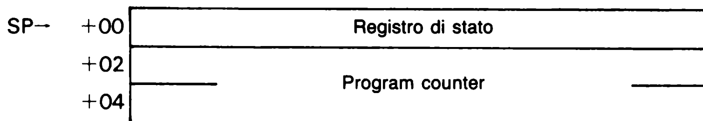


Figura 7.2 Short stack frame per il 68000/68008

Stack frame formato $0

La Figura 7.4 mostra il formato $0 dello stack frame. Il 68010, 68012 e il 68020 usano questo formato per exception dovute a interrupt, errori di formato, istruzioni TRAP, trappole di linea A e linea F, violazioni di privilegi e preistruzioni del coprocessore.

Stack frame formato $1

La Figura 7.5 mostra lo stack frame formato $1. Detto anche stack frame "throwaway", lo stack frame formato $1 è usato dal 68020 quando riceve un interrupt mentre il bit master del registro di stato è settato.

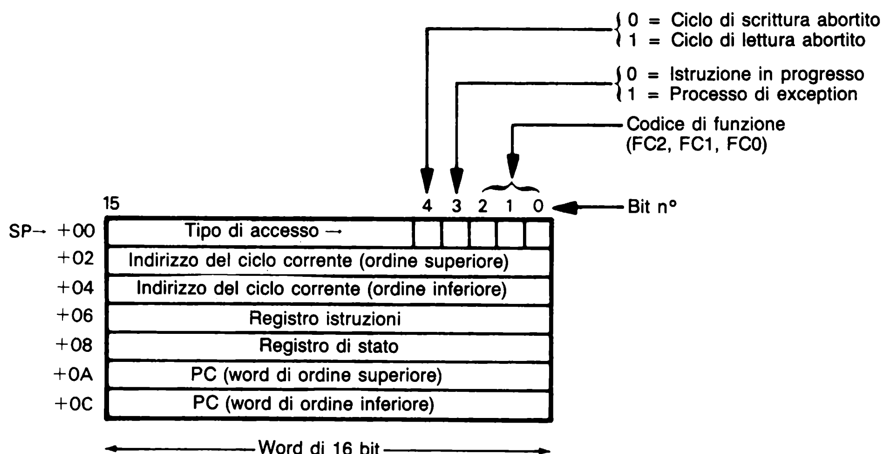


Figura 7.3 Stack frame per gli errori di bus e di indirizzamento sul 68000/68008.

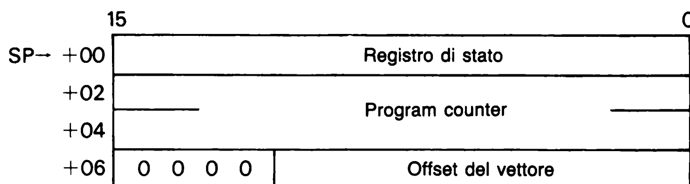


Figura 7.4 Stack frame formato $0.

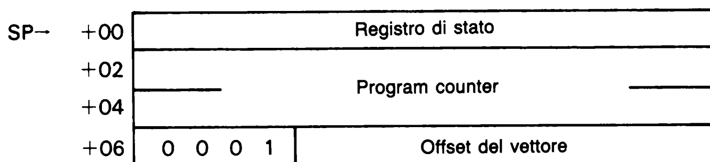


Figura 7.5 Stack frame formato $1.

Stack frame formato $2

La Figura 7.6 mostra lo stack frame formato $2. Il 68020 usa questo stack frame per exception relative alle postistruzioni del coprocessore, ovvero CHK, CHK2, cpTRAPcc, TRAPcc, TRAPV, trace e divisione per zero.

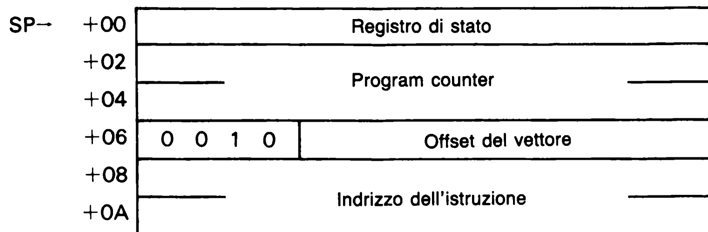


Figura 7.6 Stack frame formato $2.

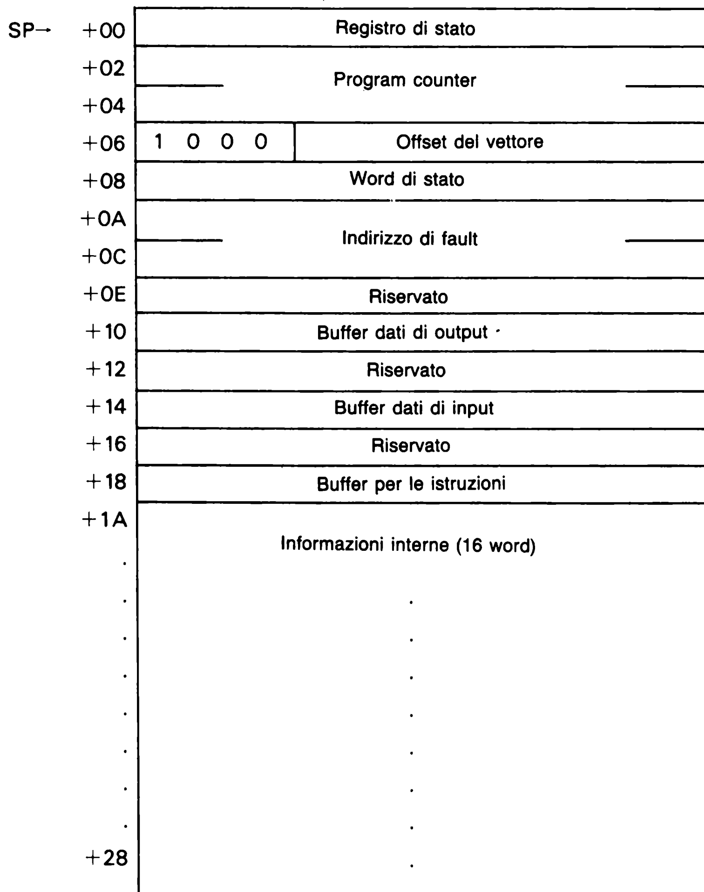


Figura 7.7 Stack frame formato $8.

Stack frame formato $8

La Figura 7.7 mostra lo stack frame formato $8. Il 68010 e il 68012 usano questo stack frame per gli errori sul bus e gli errori di indirizzamento.

Stack frame formato $9

La Figura 7.8 mostra lo stack frame formato $9. Il 68020 usa questo stack frame per exception derivanti da medio-istruzioni del coprocessore.

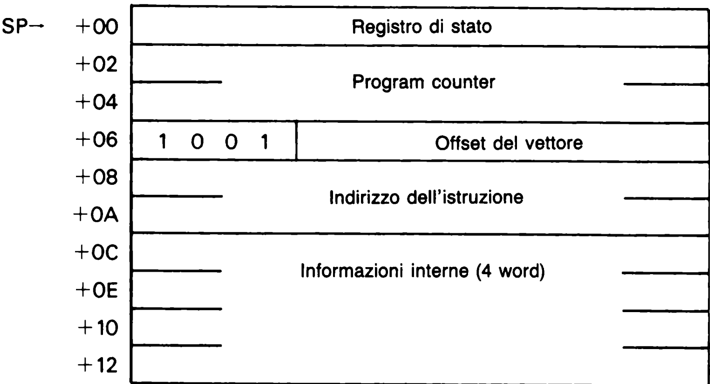


Figura 7.8 Stack frame formato $9.

Stack frame formato $A

La Figura 7.9 mostra lo stack frame formato $A. Il 68020 usa questo stack frame quando incontra un errore di ciclo sul bus che sta per eseguire un'istruzione.

Stack frame formato $B

La Figura 7.10 mostra lo stack frame formato $B. Il 68020 usa questo stack frame quando incontra un errore di ciclo sul bus durante l'esecuzione di un'istruzione. Questa condizione differisce dalla precedente in quanto vi sono molti più dati da salvare per conservare lo stato del processore.

SP→	+00	Registro di stato
	+02	Program counter
	+04	
	+06	1 0 1 0 | Offset del vettore
	+08	Informazioni interne
	+0A	Word di stato
	+0C	Pipe istruzioni stage A
	+0E	Pipe istruzioni stage B
	+10	Indirizzo di errore sul ciclo dati
	+12	
	+14	Informazioni interne
	+16	
	+18	Buffer di output dei dati
	+1A	
	+1C	Informazioni interne
	+1E	

Figura 7.9 Stack frame formato $A.

7.6 Sequenze di processo in modo exception

Ogni tipo di exception segue la medesima sequenza di eventi:

1. Effettua una copia del registro di stato e quindi setta il bit supervisore, cancella i bit di trace e, nel caso di interrupt, modifica la maschera di interrupt.

2. Determina l'elemento della tabella dei vettori leggendolo direttamente dal dispositivo che ha invocato l'interruzione oppure usando i numeri fissi associati agli altri tipi di gestione della exception.

3. Inserisce sullo stack del supervisore i dati necessari. Il tipo di dati inseriti dipende sia dal tipo di exception che dal particolare membro della famiglia 68000.

4. Carica nel program counter l'indirizzo contenuto nella tabella dei vettori e comincia l'esecuzione.

SP→	Offset	Contenuto
SP→	+00	Registro di stato
	+02	Program counter
	+04	
	+06	1 0 1 1 — Offset del vettore
	+08	Informazioni interne
	+0A	Word di stato
	+0C	Pipe istruzioni stage A
	+0E	Pipe istruzioni stage B
	+10	Indirizzo di errore sul ciclo dati
	+12	
	+14	Informazioni interne
	+16	
	+18	Buffer di output dei dati
	+1A	
	+1C	Informazioni interne
	+1E	
	+20	
	+22	
	+24	Indirizzo stage B
	+26	
	+28	Informazioni interne
	+2A	
	+2C	Buffer di input dei dati
	+2E	
	+30	Informazioni interne (22 word)
	.	.
	.	.
	.	.
	+5A	

Figura 7.10 Stack frame formato $B.

7.7 Istruzioni di tipo trap

Le istruzioni di tipo trap vengono eseguite come risultato dell'esecuzione di un'istruzione particolare. L'istruzione può attivare normalmente un processo di exception (come nel caso dell'istruzione TRAP) o può attivare il processo di exception solo quando vengono ravvisate situazioni anormali come nel caso di istruzioni di tipo TRAPcc, TRAPV, cpTRAPcc, CHK, CHK2, DIVS, DIVU, CALLM o RTM. Sul 68000 e sul 68008 il processore costruisce uno stack frame di tre word come mostrato in Figura 7.2. Sul 68010 e sul 68012 il processore costruisce uno stack frame di quattro word (formato $0) come mostrato in Figura 7.4. Il tipo di stack frame costruito dal 68020 dipende dal tipo di istruzione: una istruzione TRAP produrrà uno stack frame formato $0 (come mostrato in Figura 7.4), mentre una qualsiasi altra istruzione produrrà uno stack frame formato $2 (Figura 7.6).

7.8 Istruzioni illegali o non implementate

La gestione delle exception causate da istruzioni illegali o non implementate è simile a quella delle istruzioni di tipo trap. Il 68000 e il 68008 usano uno stack frame di tre word (Figura 7.2), mentre il 68010, 68012 e 68020 usano lo stack frame formato $0 (Figura 7.4).
Da notare è la distinzione fra istruzioni illegali e istruzioni non implementate; un codice istruzioni qualsiasi che incominci con le configurazioni di bit 1010 (linea A) o 1111 (linea F) viene considerato come istruzione non implementata. Le istruzioni di linea A o di linea F posseggono i loro elementi nella tabella dei vettori: per questo motivo è conveniente emulare le istruzioni non implementate.
Il 68020 considera le istruzioni della linea F come potenziali istruzioni per il coprocessore; dopo aver deciso che non è in grado di eseguire l'istruzione da solo, il processore esegue un ciclo di bus nello spazio di CPU (vedi Figura 5.6) codificando le informazioni di identificazione del coprocessore sul bus indirizzi. Se nessun coprocessore risponde, la logica esterna segnala un errore sul bus e il processore inizia il processo di exeption corrispondente all'elemento di linea F della tabella dei vettori.
Se state emulando una istruzione non implementata via software noterete che il program counter memorizzato nello stack frame punta all'istruzione illegale. Quando avrete terminato di emulare le istruzioni, dovete ricordarvi di modificare il PC memorizzato sullo stack in modo tale da farlo puntare all'istruzione che segue l'istruzione illegale.

7.9 Errore di indirizzamento

Come abbiamo già stabilito in precedenza, le istruzioni devono trovarsi a indirizzi pari. Se il processore tenta di accedere a un'istruzione che si trova a un indirizzo dispari viene iniziato immediatamente un processo di exception. Il 68000 e il 68008 costruiscono uno stack frame di 7 word come mostrato in Figura 7.3. Il 68010 e il 68012 costruiscono uno stack frame formato $8 (Figura 7.7) e il 68020 costruisce uno stack frame formato $A (Figura 7.9). Notate che se l'errore di indirizzamento viene incontrato durante il processo di exception causato da un errore sul bus, da un errore di indirizzamento o reset il processore entra in uno stato di halt e deve essere resettato dall'esterno.

7.10 Tracing

Tutti i membri della famiglia 68000 implementano il trace a passo singolo: in questa modalità il processore dà inizio al processo di exception dopo l'esecuzione di ogni istruzione. Quando il bit di trace (dal 68000 al 68012) o il bit T1 (68020) del registro di stato vengono settati il processore comincia a operare in modo trace dopo ogni istruzione.

Per evitare che i processi di exception divengano ricorsivi, una delle prime cose che viene fatta dal processore durante il processo di exception è quella di pulire i bit di trace. Il 68000 e il 68008 costruiscono uno stack frame di tre word (Figura 7.2); il 68010 e il 68012 costruiscono uno stack frame formato $0 (Figura 7.4) e il 68020 costruisce uno stack frame formato $2 (Figura 7.6).

Il 68020 permette una forma di tracing più selettiva rispetto ai suoi predecessori: il tracing sui cambi di flusso. Quando il bit T0 del registro di stato viene settato, il processore esegue un'istruzione in grado di interrompere l'esecuzione sequenziale delle istruzioni. Queste istruzioni comprendono Bcc, JSR, BSR e così via e dopo che sono state eseguite inducono automaticamente il processo di exception.

Notate che il tentativo di eseguire un'istruzione illegale o non implementata *non* provoca una trace. Ciò è importante nel caso in cui si voglia emulare delle istruzioni; in questo caso la routine di emulazione dovrebbe controllare il bit di trace del registro di stato memorizzato sullo stack prima di cedere il controllo al programma chiamante; se questo bit è settato la routine di emulazione dovrebbe emulare anche il trace.

7.11 Breakpoint

In alcuni sistemi potrebbe essere conveniente introdurre dei breakpoint in un programma, ossia dei punti in cui l'esecuzione del programma viene momentaneamente interrotta. Per essere veramente utile in un sistema di emulazione hardware, il processore deve avere la possibilità di comunicare all'hardware esterno che è stato raggiunto un breakpoint. Sul 68000 e sul 68008 un programma può inserire come breakpoint un'istruzione illegale: l'hardware esterno può monitorare le linee di indirizzamento cercando un accesso al vettore relativo alle istruizoni illegali nella tabella dei vettori.
Sul 68010, 68012 e 68020 questo meccanismo non è più efficace dal momento che questi processori posseggono il registro base dei vettori (VBR), che permette a un programma di cambiare dinamicamente la locazione fisica della tabella dei vettori. Questi processori definiscono uno speciale set di istruzioni note come istruzioni di tipo "breakpoint". Queste istruzioni hanno i pattern da $4848 a $484F.
Sul 68010 e sul 68012 il processore esegue dapprima un "ciclo di bus per i breakpoint". Come mostrato in Tabella 5.2 esso viene classificato come ciclo di bus sullo spazio di CPU. Dopo aver chiamato questo ciclo, il processore esegue una trap attraverso il vettore delle istruzioni illegali e prosegue normalmente l'esecuzione delle istruzioni illegali.
Sul 68020, come sul 68010 e sul 68012, il processore non solo esegue un ciclo di bus per i breakpoint ma, durante questo ciclo, esegue anche una lettura dallo spazio di CPU. Se questa lettura viene terminata da un errore di bus, il processore verrà intrappolato nel vettore relativo alle istruzioni illegali. In ogni caso l'hardware esterno può inserire una nuova istruzione sul bus dati e terminare il ciclo con il segnale DSACKx. In questo caso il processore sostituisce l'istruzione di breakpoint (nella pipeline) con l'istruzione sul bus dati e riprende l'esecuzione a partire da quest'ultima istruzione.

7.12 Errori di formato

Il 68020 ha tre istruzioni che eseguono un controllo sui dati: la chiamata a un modulo (CALLM), il ritorno da modulo (RTM) e il restore del coprocessore (cpRESTORE). Queste istruzioni si aspettano di trovare dei dati ben precisi all'interno degli stack frame, dei descrittori di modulo e dei campi formato. Se i dati contenuti in queste locazioni non corrispondono a quanto si aspettano queste istruzioni, esse verranno intrappolate nel vettore relativo all'exception causata da un errore di formato, usando uno stack frame formato $0 (Figura 7.4).
L'Appendice B tratterà più in dettaglio l'interfaccia con il coprocessore.

7.13 Interrupt

I dispositivi esterni possono interrompere il flusso normale delle istruzioni del processore attivando una o più linee di richiesta di interrupt (IPL0-IPL2). Il processore riconosce tale richiesta nello spazio di tempo intercorrente fra l'esecuzione delle varie istruzioni e, nel caso del 68020, durante certe istruzioni del coprocessore. Al ricevimento della richiesta di interrupt, il processore confronta il valore binario indicato da IPL0-IPL2 con la maschera di interrupt del registro di stato. Se IPL0-IPL2 mostrano una priorità più alta rispetto alla maschera o se IPL0-IPL2 sono tutte attivate ("interrupt non mascherabile"), il processore inizia il processo di exception. In caso contrario il processore ignora la richiesta e continua con la normale esecuzione.
Se il processore desidera riconoscere l'interrupt, esso esegue un ciclo di bus sullo spazio di CPU emettendo sul bus indirizzi la priorità dell'interrupt su A1-A3 (vedi Tabella 5.2). Il dispositivo può scegliere se spedire al processore il numero dell'elemento della tabella dei vettori attivando DTACK (o DSACK0/DSACK1) e caricando il numero del vettore sul bus dati.
Il dispositivo potrebbe però richiedere al processore di usare l'elemento relativo all'autovettore. Sul 68000-68012 esso deve attivare VPA; sul 68020 esso deve attivare AVEC. Il processore verrà quindi intrappolato nell'elemento dell'autovettore associato al livello di interrupt indicato.
Se nessun dispositivo risponde al ciclo di accettazione dell'interrupt, l'hardware esterno deve attivare BERR. Il processore verrà quindi intrappolato nell'indirizzo del vettore relativo agli interrupt spuri.
La maggior parte delle nuove periferiche costruite per il 68000 ha la possibilità di programmare il numero dei vari vettori di interrupt che essi spediscono al processore al momento dell'accettazione dell'interrupt. Se non sono stati inizializzati via software questi dispositivi spediscono di default il vettore $0F. Questo vettore viene definito come elemento non inizializzato e, sulla precedente documentazione, veniva indicato come "riservato".
Lo stack frame creato per le exception relative agli interrupt varia a seconda dei vari processori. Sul 68000 e sul 68008 il processore costruisce uno stack frame di tre word (Figura 7.2). Sul 68010 e sul 68012 il processore costruisce lo stack frame formato $0 (Figura 7.4). Sul 68020 lo stack frame dipende da ciò che il processore sta eseguendo prima di essere interrotto: se si trova fra due istruzioni, esso inserirà sullo stack supervisore uno stack frame formato $0 (Figura 7.4); se è stato interrotto durante un'istruzione di coprocessore, esso costruirà uno stack frame formato $9 (Figura 7.8).
La locazione dello stack frame sul 68020 dipende dallo stato del bit M del registro di stato. Se questo bit è basso, il processore costruisce lo stack frame sullo stack dedicato agli interrupt. Se invece questo bit è settato il processore costruisce lo stack frame sullo stack principale e poi costruisce uno stack frame "temporaneo" (formato $1) sullo stack relativo agli interrupt.

7.14 Errori sul bus

In risposta a una richiesta di lettura o scrittura, la logica esterna può mettere a disposizione o incamerare il dato e attivare $\overline{DTACK}$ ($\overline{DSACK0}/\overline{DSACK1}$) oppure può rispondere con un segnale di errore sul bus ($\overline{BERR}$). La logica esterna può attivare $\overline{BERR}$ in quanto la memoria non è fisicamente presente o perché la zona di memoria richiesta non appartiene allo spazio di indirizzamento accessibile dall'utente. Per evitare inconvenienti di questo tipo spesso i sistemi usano unità di gestione della memoria (MMU o Memory Management Unit) per determinare se un particolare slot di memoria è disponibile o meno.
Sul 68000 e sul 68008, al momento della ricezione di un segnale di errore sul bus, il processore costruisce sullo stack uno stack frame di sette word (Figura 7.3). Notate che il contatore di programma memorizzato sullo stack fornisce l'indirizzo della word che segue la word che ha provocato l'errore sul bus.

MEMORIA VIRTUALE

Il 68010, il 68012 e il 68020 implementano un meccanismo di "memoria virtuale". In un sistema che implementa questo tipo di memoria, lo spazio di indirizzamento totale dei task residenti nel sistema può essere molto più grande della memoria fisicamente disponibile nel sistema. Il sistema operativo può scegliere di memorizzare certi dati o un certo codice su memoria secondaria (ad esempio un disco) mentre altre sezioni del codice sono in esecuzione. Quando viene richiesto il codice scaricato su disco, il sistema operativo deve essere in grado di copiare velocemente su disco i dati che al momento non sono necessari e leggere i dati e il codice richiesti.
Per implementare la memoria virtuale in modo efficace, il processore deve essere in grado di riconoscere che la zona di memoria cui il programma vuole accedere non si trova nella memoria fisica, rintracciare dal disco la zona di memoria desiderata, e continuare l'esecuzione del programma. La logica esterna può accorgersi che la memoria richiesta non esiste: in questo caso essa attiva $\overline{BERR}$. Il 68000 e il 68008 non sono comunque in grado di salvare abbastanza informazioni sullo stack per permettere al processore di rieseguire l'istruzione una volta che il dato viene caricato in memoria.
Il 68010, il 68012 e il il 68020 creano stack frame più complessi quando iniziano il processo di exception relativo agli errori sul bus. Il 68010 e il 68012 creano stack frame formato $8 (Figura 7.7) e il 68020 crea uno stack frame formato $A (per errori di bus durante la fase di fetch di un'istruzione (Figura 7.9)) o uno stack frame formato $B (per errori sul bus occorsi durante l'esecuzione delle istruzioni (Figura 7.10)).

I dati in più che vengono memorizzati contengono informazioni relative allo stato interno del processore. Come parte delle routine di exception, il processore può valutare le informazioni memorizzate sullo stack e determinare se l'errore era dovuto a una richiesta di dati memorizzati su disco. Se è così, esso può trasferire i dati fuori dalla memoria e inserirvi i dati richiesti. Dopo l'aggiornamento dei registri di gestione della memoria, il sistema operativo può permettere che il programma che ha fallito il tentativo possa riprovare più avanti.
Per fare ciò il processore esegue un'istruzione di ritorno da exception (RTE): questa istruzione controlla il formato dello stack frame e determina che il dato memorizzato sullo stack è il risultato di un errore sul bus: a questo punto il processore reintroduce i dati sullo stack e riprende l'esecuzione del programma dal punto indicato dal program counter. Notate che il punto indicato dallo stack pointer potrebbe trovarsi nel mezzo o al principio di un'istruzione.

DOPPIO BUS FAULT

In tutti i processori 68000, se durante la fase di fetch o la fase di memorizzazione sullo stack di un vettore il processore incontra un altro bus fault, esso entra nello stato di halt. Questa situazione è chiamata "doppio bus fault" e l'unico modo per uscirne è un reset esterno.

7.15 Reset

L'exception relativa al reset è unica fra i vari tipi di exception in quanto essa non scrive alcuna informazione sullo stack. Quando viene attivato un reset esterno vengono effettuate le operazioni seguenti:

1. Le operazioni correnti vengono interrotte.

2. Nel registro di stato viene settato il bit master, viene cancellato il bit supervisore (solo sul 68020), i bit di trace vengono cancellati, la maschera di interrupt viene settata al livello 7, il registro base dei vettori viene settato a $0000 (68010-68020) e il registro di controllo della cache viene cancellato (68020).

3. Il processore carica lo stack pointer relativo allo stack supervisore (ISP sul 68020) con la long word che si trova all'offset $0000 della tabella dei vettori e carica nel program counter la long word a un offset pari a $0004. Notate la fase di fetch di questo elemento del vettore avviene solo nello

spazio dei programmi di supervisore, mentre tutti gli altri vettori di exception avvengono nell'area dati del supervisore.

4. Il processore attiva $\overline{\text{RESET}}$ per resettare i dispositivi esterni e riprende l'esecuzione a partire dall'istruzione il cui indirizzo è contenuto nel program counter.

Notate che, tranne che per i registri specificati in precedenza, tutti gli altri registri risulteranno a questo punto indefiniti.

A

Codice oggetto delle istruzioni

OR immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

OR immediato con il CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	Byte dati							

OR immediato con SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
Word dati															

CMP2 (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	Dimensione		0	1	1	Indirizzo effettivo					
										Modalità			Registro		
I/D	Registro			0	0	0	0	0	0	0	0	0	0	0	0

Campo dimensione: 00 = byte 01 = word 10 = long

CHK2 (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	Dimensione		0	1	1	Indirizzo effettivo					
										Modalità			Registro		
I/D	Registro			1	0	0	0	0	0	0	0	0	0	0	0

Campo dimensione: 00 = byte 01 = word 10 = long

Bit dinamico

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Registro dati			1	Tipo		Indirizzo effettivo					
										Modalità			Registro		

Campo tipo: 00 = TST 10 = CLR
01 = CHG 11 = SET

MOVEP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Registro dati			Modalità operativa			0	0	1	Registro indirizzi		

Campo modalità operativa: 100 = trasferimento di una word da memoria a registro
101 = trasferimento di una long word da memoria a registro
110 = trasferimento di una word da registro a memoria
111 = trasferimento di una long word da registro a memoria

AND immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

AND immediato con CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	Byte dati							

AND immediato con SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0
Word dati															

SUB immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

ADD immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione 00 = byte 01 = word 10 = long

RTM (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	0	0	D/I	Registro		

CALLM (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	Indirizzo effettivo					
										Modalità			Registro		
0	0	0	0	0	0	0	0	Contatore degli argomenti							

CAS (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	Dimensione		0	1	1	Indirizzo effettivo					
										Modalità			Registro		
0	0	0	0	0	0	0	Du			0	0	0	Dc		

Campo dimensione: 00 = byte 10 = word 11 = long

CAS2 (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	Dimensione		0	1	1	1	1	1	1	0	0
D/I	Registro 1			0	0	0	Du1			0	0	0	Dc1		
D/I	Registro 2			0	0	0	Du2			0	0	0	Dc2		

Campo dimensione: 01 = byte 10 = word 11 = long

Bit statico

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										Indirizzo effettivo					
0	0	0	0	1	0	0	0	Tipo		Modalità			Registro		

Campo tipo: 00 = TST 10 = CLR
01 = CHG 11 = SET

EOR immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										Indirizzo effettivo					
0	0	0	0	1	0	1	0	Dimensione		Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

EOR immediato con CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	Byte dati							

EOR immediato con SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0
Word dati															

CMP immediato

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										Indirizzo effettivo					
0	0	0	0	1	1	0	0	Dimensione		Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

MOVES

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		
I/D	Registro			dr	0	0	0	0	0	0	0	0	0	0	0

Campo dr: 0 = da EA a registro
1 = da registro a EA

MOVE Byte

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	Destinazione						Sorgente					
				Registro			Modalità			Modalità			Registro		

Notare le locazioni dei registri e delle modalità

MOVEA Long

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Registro destinazione			0	0	1	Sorgente					
										Modalità			Registro		

MOVE Long

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Destinazione						Sorgente					
				Registro			Modalità			Modalità			Registro		

Notare le locazioni dei registri e delle modalità

MOVEA Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	Registro destinazione			0	0	1	Sorgente					
										Modalità			Registro		

MOVE Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	Destinazione						Sorgente					
				Registro			Modalità			Modalità			Registro		

Notare le locazioni dei registri e delle modalità

NEGX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

MOVE da SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	Indirizzo effettivo					
										Modalità			Registro		

CHK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Registro dati			Dimensione		0	Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 10 = long word (MC68020)
11 = word

LEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Registro indirizzi			1	1	1	Indirizzo effettivo					
										Modalità			Registro		

CLR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

MOVE da CCR (MC68010)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	1	1	Indirizzo effettivo					
										Modalità			Registro		

NEG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

MOVE verso CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	1	1	Indirizzo effettivo: Modalità			Registro		

NOT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	Dimensione		Indirizzo effettivo: Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

MOVE verso SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	1	1	Indirizzo effettivo: Modalità			Registro		

NBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	Indirizzo effettivo: Modalità			Registro		

LINK Long word (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	0	0	1	Registro dati		
Spiazzamento di ordine superiore															
Spiazzamento di ordine inferiore															

SWAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	0	Registro dati		

BKPT (MC68010)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	1	Vettore		

PEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	Indirizzo effettivo					
										Modalità			Registro		

EXT/EXTB (EXTB-MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	Tipo			0	0	0	Registro dati		

Campo tipo: 010 = word di extend 011 = long di extend 111 = long di extended byte (MC68020)

MOVEM Registri verso EA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	Dim.	Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 0 = trasferimento di una word 1 = trasferimento di una long

TST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 00 = byte 01 = word 10 = long

TAS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	Indirizzo effettivo					
										Modalità			Registro		

ILLEGAL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0

MULS/MULU Long (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	Indirizzo effettivo					
										Modalità			Registro		
0	Dl			Tipo	Dim.	0	0	0	0	0	0	Dh			

Campo tipo: 0 = MULU
1 = MULS

Campo dimensione: 0 = prodotto fra long word
1 = prodotto fra quad word

DIVS/DIVU Long (MC68020)
DIVUL/DIVSL (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	1	Indirizzo effettivo					
										Modalità			Registro		
0	Dq			Tipo	Dim.	0	0	0	0	0	0	0	Dr		

Campo tipo: 0 = DIVU
1 = DIVS

Campo dimensione: 0 = dividendo long word
1 = dividendo quad word

MOVEM EA verso registri

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	1	Dim.	Indirizzo effettivo					
										Modalità			Registro		

Campo dimensione: 0 = trasferimento di una word 1 = trasferimento di una long

TRAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	Vettore			

LINK WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Registro indirizzi		

UNLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Registro indirizzi		

MOVE verso USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	0	Registro indirizzi		

MOVE da USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	1	Registro indirizzi		

RESET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

STOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0

RTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

RTD (MC68010)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	0

RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

TRAPV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0

RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1

MOVEC (MC68010)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	dr
I/D	Registro			Registro di controllo											

Campo dr: 0 = registro di controllo a registro generale
1 = registro generale a registro di controllo

Campo del registro di controllo: $000 = SFC | $001 = DFC | $002 = CACR (MC68020) | $800 = USP | $801 = VBR | $802 = CAAR (MC68020) | $803 = MSP (MC68020) | $804 = ISP (MC68020)

JSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	Indirizzo effettivo					
										Modalità			Registro		

JMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	1	Indirizzo effettivo					
										Modalità			Registro		

ADDQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Dato			0	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dati: Tre bit di dati immediati, 0, 1-7 rappresentanti un range di 8 numeri, (da 1 a 7)
Campo dimensione: 00 = byte 01 = word 10 = long

Scc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condizione				1	1	Indirizzo effettivo					
										Modalità			Registro		

DBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condizione				1	1	0	0	1	Registro dati		

TRAPcc (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condizione				1	1	1	1	1	Modalità		
Operando															

Campo modalità: 010 = operando di tipo word
011 = operando di tipo long word 100 = nessun operando

SUBQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Dato			1	Dimensione		Indirizzo effettivo					
										Modalità			Registro		

Campo dati: Tre bit di dati immediati, 0, 1-7 rappresentanti un range di 8 numeri (da 1 a 7)
Campo dimensione: 00 = byte 01 = word 10 = long

Bcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	Condizione				Spiazzamento di 8 bit							
Spiazzamento di 16 bit se lo spiazzamento di 8 bit = $00															
Spiazzamento di 32 bit se lo spiazzamento di 8 bit = $FF															

BRA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	Spiazzamento di 8 bit							
Spiazzamento di 16 bit se lo spiazzamento di 8 bit = $00															
Spiazzamento di 32 bit se lo spiazzamento di 8 bit = $FF															

BSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	Spiazzamento di 8 bit							
Spiazzamento di 16 bit se lo spiazzamento di 8 bit = $00															
Spiazzamento di 32 bit se lo spiazzamento di 8 bit = $FF															

MOVEQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	Registro dati			0	Dati							

Campo dati: Il dato è esteso con segno a un operando di tipo long e tutti i 32 bit sono trasferiti nel registro dati

OR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										Indirizzo effettivo					
1	0	0	0	Registro dati			Modalità operativa			Modalità			Registro		

Campo della modalità operativa:

Byte	Word	Long	Operazione
000	001	010	(<ie>)v(<Dn>)→<Dn>
100	101	110	(<Dn>)v(<ie>)→<ie>

DIVU/DIVS Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										Indirizzo effettivo					
1	0	0	0	Registro dati			Tipo	1	1	Modalità			Registro		

Campo tipo: 0 = DIVU 1 = DIVS

SBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Registro destinazione*			1	0	0	0	0	R/M	Registro sorgente*		

Campo R/M: 0 = da registro dati a registro dati
1 = da memoria a memoria
*R/M = 0 specifica un registro dati
R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

PACK (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Registro destinazione*			1	0	1	0	0	R/M	Registro sorgente*		
Estensione: Aggiustamento a 16 bit															

Campo R/M: 0 = da registro dati a registro dati
1 = da memoria a memoria
*R/M = 0 specifica un registro dati
R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

UNPK (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Registro destinazione*			1	1	0	0	0	R/M	Registro sorgente*		

Campo R/M: 0 = da registro dati a registro dati
1 = da memoria a memoria
*R/M = 0 specifica un registro dati
R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

SUB

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	0	0	1	Registro dati	Modalità operativa	Modalità	Registro

Campo della modalità operativa:	Byte	Word	Long	Operazione
	000	001	010	(<ie>)−(<Dn>)→<Dn>
	100	101	110	(<Dn>)−(<ie>)→<ie>

SUBA

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	0	0	1	Registro dati	Modalità operativa	Modalità	Registro

Campo della modalità operativa:	Word	Long	Operazione
	011	111	(<ie>)−(<In>)→<In>

SUBX

15	14	13	12	11 10 9	8	7 6	5	4	3	2 1 0
1	0	0	1	Registro destinazione*	1	Dimensione	0	0	R/M	Registro sorgente*

Campo dimensione: 00 = byte 01 = word 10 = long
Campo R/M: 0 = da registro dati a registro dati 1 = da memoria a memoria
*R/M = 0 specifica un registro dati
R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

CMP

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	0	1	1	Registro dati	Modalità operativa	Modalità	Registro

Campo relativo alla modalità operativa:	Byte	Word	Long	Operazione
	000	001	010	(<Dn>)−(<ie>)

CMPA

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	0	1	1	Registro dati	Modalità operativa	Modalità	Registro

Campo relativo alla modalità operativa:	Word	Long	Operazione
	011	111	(<In>)−(<ie>)

EOR

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	0	1	1	Registro dati	Modalità operativa	Modalità	Registro

Campo relativo alla modalità operativa:

Byte	Word	Long	Operazione
100	101	110	(<ie>)v(<Dn>)→<ie>

CMPM

15	14	13	12	11 10 9	8	7 6	5	4	3	2 1 0
1	0	1	1	Registro destinazione	1	Dimensione	0	0	1	Registro sorgente

Campo dimensione: 00 = byte 01 = word 10 = long

AND

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
						Indirizzo effettivo	
1	1	0	0	Registro dati	Modalità operativa	Modalità	Registro

Campo relativo alla modalità operativa:

Byte	Word	Long	Operazione
000	001	010	(<ie>)∧(<Dn>)→<Dn>
100	101	110	(<Dn>)∧(<ie>)→<ie>

MULU Word
MULS Word

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
								Indirizzo effettivo	
1	1	0	0	Registro dati	0	1	Tipo	Modalità	Registro

Campo tipo: 0 = MULU 1 = MULS

ABCD

15	14	13	12	11 10 9	8	7	6	5	4	3	2 1 0
1	1	0	0	Registro destinazione*	1	0	0	0	0	R/M	Registro sorgente*

Campo R/M: 0 = da registro dati a registro dati 1 = da memoria a memoria
*R/M = 0 specifica un registro dati
R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

EXC Registri dati

15	14	13	12	11 10 9	8	7	6	5	4	3	2 1 0
1	1	0	0	Registro dati	1	0	1	0	0	0	Registro dati

EXC Registri indirizzi

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Registro indirizzi			1	0	1	0	0	1	Registro indirizzi		

EXC Registro dati e registro indirizzi

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Registro dati			1	1	0	0	0	1	Registro indirizzi		

ADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Registro dati			Modalità operativa			Indirizzo effettivo					
										Modalità			Registro		

Campo relativo alla modalità operativa:	Byte	Word	Long	Operazione
	000	001	010	(<ie>)+(<Dn>)→<Dn>
	100	101	110	(<Dn>)+(<ie>)→<ie>

ADDA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Registro indirizzi			Modalità operativa			Indirizzo effettivo					
										Modalità			Registro		

Campo relativo alla modalità operativa:	Word	Long	Operazione
	011	111	(<ie>)+(<Dn>)→<Dn>

ADDX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Registro destinazione*			1	Dimensione		0	0	R/M	Registro sorgente*		

Campo dimensione: 00 = byte 01 = word 10 = long

Campo R/M: 0 = da registro dati a registro dati 1 = da memoria a memoria

*R/M = 0 specifica un registro dati

R/M = 1 specifica un registro indirizzi per il modo di indirizzamento con predecremento

SHIFT/ROTATE - Registro

15	14	13	12	11 10 9	8	7 6	5	4 3	2 1 0
1	1	1	0	Contatore/registro	dr	Dimensione	i/r	Tipo	Registro dati

Campo contatore/registro: campo i/r = 0 specifica il contatore relativo allo shift
campo i/r = 1 specifica un registro dati che contiene il contatore relativo allo shift

Campo dr: 0 = destra 1 = sinistra
Campo dimensione: 00 = byte 01 = word 10 = long
Campo i/r: 0 = shift count immediato 1 = shift count del registro
Campo tipo: 00 = shift aritmetico 10 = rotazione con estensione
01 = shift logico 11 = rotazione

SHIFT/ROTATE - Memoria

15	14	13	12	11	10 9	8	7	6	5 4 3	2 1 0
									Indirizzo effettivo	
1	1	1	0	0	Tipo	dr	1	1	Modalità	Registro

Campo tipo: 00 = shift artmetico 01 = shift logico 10 = rotazione con estensione
11 = rotazione
0 = destra 1 = sinistra

Campi di bit (MC68020)

15	14 13 12	11	10 9 8 7 6	5	4 3 2 1 0
1 1 1 0		1	Tipo (10 9 8) 1 1	Indirizzo effettivo: Modalità	Registro
0	Registro	Do	Offset	Dw	Ampiezza

Campo tipo: 000 = BFTST 100 = BFCLR
001 = BFEXTU 101 = BFFFO
010 = BFCHG 110 = BFSET
011 = BFEXTS 111 = BFINS
Il campo registro è 000 per BFTST, BFCHG, BFCLR e BFSET
Campo Do: 0 = offset immediato 1 = offset nel registro dati
Campo Dw: 0 = ampiezza immediata 1 = ampiezza nel registro dati

ISTRUZIONI DEL COPROCESSORE

cpGEN (MC68020)

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
								Indirizzo effettivo	
1	1	1	1	Cp-Id	0	0	0	Modalità	Registro
Command word dipendente dal coprocessore									

cpScc (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			0	0	1	Indirizzo effettivo					
										Modalità			Registro		
0	0	0	0	0	0	0	0	0	0	Condizione del coprocessore					

cpDBcc (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			0	0	1	0	0	1	Registro		
0	0	0	0	0	0	0	0	0	0	Condizione del coprocessore					
Spiazzamento															

cpTRAPcc (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			0	0	1	1	1	1	Modalità		
0	0	0	0	0	0	0	0	0	0	Condizione del coprocessore					
Operando															

Campo modalità: 010 = operando word 011 = operando long word 100 = nessuno spiazzamento

cpBcc (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			0	1	Dim.	Condizione del coprocessore					
Spiazzamento															

Campo dimensione: 0 = spiazzamento di 1 word 1 = spiazzamento di una long word

cpSAVE (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			1	0	0	Indirizzo effettivo					
										Modalità			Registro		

cpRESTORE (MC68020)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Cp-Id			1	0	1	Indirizzo effettivo					
										Modalità			Registro		

PRIMITIVE DEL COPROCESSORE (MC68020)

Busy
(occupato)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	PC	1	0	0	1	0	0	0	0	0	0	0	0	0	0

TRANSFER MULTIPLE COPROCESSOR REGISTERS
(trasferimento di registri del coprocessore)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	0	0	0	1	Lunghezza							

TRANSFER STATUS REGISTER AND SCANPC
(trasferimento del registro di stato e dello SCANPC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	0	0	1	SP	0	0	0	0	0	0	0	0

SUPERVISOR CHECK
(controllo dello stato di supervisore)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	0	0	1	0	0	0	0	0	0	0	0	0	0

TAKE ADDRESS AND TRANSFER DATA
(lettura dell'indirizzo e trasferimento dei dati)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	0	1	0	1	Lunghezza							

TRANSFER MULTIPLE MAIN PROCESSOR REGISTERS
(trasferimento di alcuni registri del processore principale)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	0	1	1	0	0	0	0	0	0	0	0	0

TRANSFER OPERATION WORD
(trasferimento della word relativa all'operazione)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	0	0	1	1	1	0	0	0	0	0	0	0	0

NULL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	0	1	0	0	IA	0	0	0	0	0	0	PF	TF

EVALUATE EFFECTIVE ADDRESS AND TRANSFER ADDRESS
(valutazione e trasferimento dell'indirizzo effettivo)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	0	1	0	1	0	0	0	0	0	0	0	0	0

TRANSFER SINGLE MAIN PROCESSOR REGISTER
(trasferimento di un singolo registro del processore)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	1	1	0	0	0	0	0	0	D/I	Registro		

TRANSFER MAIN PROCESSOR CONTROL REGISTER
(trasferimento del registro di controllo del processore)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	0	1	1	0	1	0	0	0	0	0	0	0	0

TRANSFER TO/FROM TOP OF STACK
(trasferimento da/verso la cima dello stack)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	1	1	1	0	Lunghezza								

TRANSFER INSTRUCTION STREAM
(trasferimento di un flusso di istruzioni)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	0	0	1	1	1	1	Lunghezza							

EVALUATE EFFECTIVE ADDRESS AND TRANSFER DATA
(valutazione dell'indirizzo effettivo e trasferimento di dati)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	dr	1	0	IE valido			Lunghezza							

TAKE PRE-INSTRUCTION EXCEPTION
(prendi l'exception relativa alla preistruzione)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PC	0	1	1	1	0	0	Offset del vettore							

TAKE MID-INSTRUCTION EXCEPTION
(prendi l'exception relativa alla medio-istruzione)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PC	0	1	1	1	0	1	Offset del vettore							

TAKE POST-INSTRUCTION EXCEPTION
(prendi l'exception relativa alla postistruzione)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PC	0	1	1	1	1	0	Offset del vettore							

WRITE TO PREVIOUSLY EVALUATED EFFECTIVE ADDRESS
(scrittura all'indirizzo effettivo valutato in precedenza)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CA	PC	1	0	0	0	0	0	Lunghezza							

B

Interfacce per i processori 68000

Nei due paragrafi di questa appendice illustreremo due caratteristiche dei processori 68000: l'interfacciamento con periferiche della famiglia 6800 e l'interfacciamento con un qualsiasi coprocessore.

B.1 Interfacce con la famiglia 6800

Quando la Motorola mise sul mercato il 68000 erano disponibili poche periferiche che implementavano il bus dati a 16 bit e il controllo asincrono del bus. Vi era comunque un buon numero di periferiche disponibili per un elemento della famiglia di processori precedente, il 6800. Al fine di permettere ai progettisti di sviluppare da subito sistemi basati sul 68000 senza dover aspettare la realizzazione di periferiche adatte, la Motorola aggiunse tre segnali al 68000 (e anche a 68008, 68010 e 68012) in modo da permettere a questi processori di interfacciarsi con i membri della famiglia 6800.
Come ricorderete senz'altro, i processori 68000 usano un handshaking asincrono per controllare il trasferimento dei dati da e verso periferiche. La famiglia 6800 d'altro canto richiede un trasferimento dati sincrono. I segnali tipici della famiglia 6800 che si trovano anche sul 68000 permettono al processore di simulare l'handshaking sincrono.
I segnali richiesti sono quello di indirizzo di memoria valido ($\overline{VMA}$ o Valid Memory Address), quello di indirizzo di periferica valido ($\overline{VPA}$ o Valid Peripheral Address) e il segnale di abilitazione (E o Enable). La Figura B.1 illustra la temporizzazione dei cicli sincroni di lettura e scrittura. Dopo che il 68000 ha mandato in output l'indirizzo sul bus indirizzi e ha attivato lo stro-

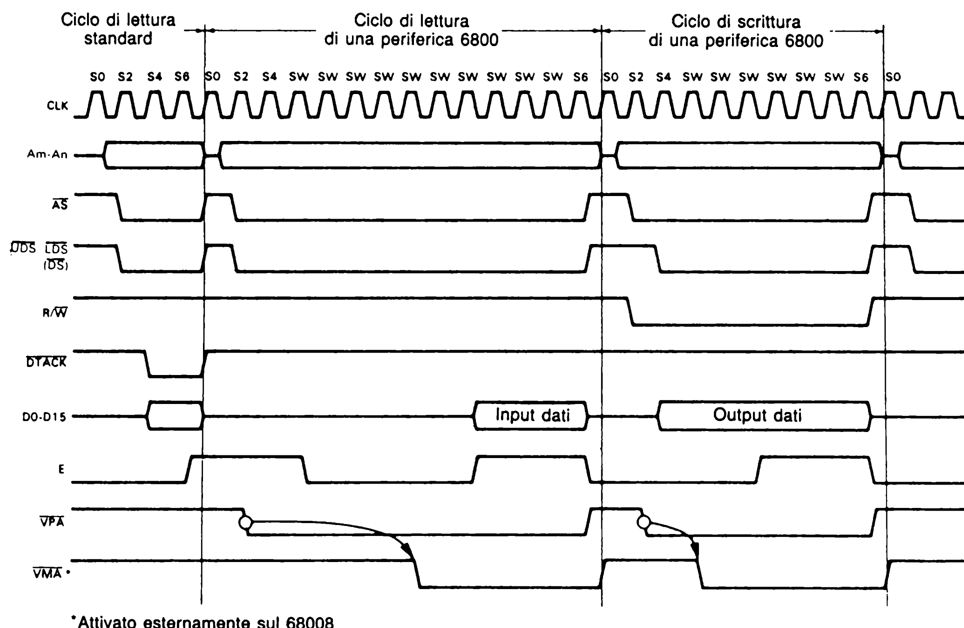


Figura B.1 Temporizzazione della lettura/scrittura sincrone per le periferiche di tipo 68000.

be indirizzi, tocca alla logica esterna decodificare l'informazione presente sulle linee degli indirizzi. Se è necessario accedere a un dispositivo della famiglia 6800, la logica esterna deve attivare in input il segnale $\overline{\text{VPA}}$. Questa operazione comunica alla periferica di continuare il trasferimento dati usando il segnale di temporizzazione sincrono (sfruttando il segnale E come clock). Durante un ciclo di lettura, la periferica di tipo 6800 deve inserire i dati sul bus dati quando il segnale E è alto: il fronte di discesa di E indica che il trasferimento dati è stato completato (invece che $\overline{\text{DTACK}}$ come nel caso di trasferimento asincrono). Il 68000 quindi procede al completamento del ciclo nel modo normale negando i segnali di strobe e rimettendo il bus indirizzi in uno stato ad alta impedenza.

TEMPORIZZAZIONE DELLE OPERAZIONI DI LETTURA/SCRITTURA DEL 6800

Noterete in Figura B.1 che esiste una differenza nel numero totale di cicli di clock per le operazioni di lettura e scrittura: da ciò non si deve però dedurre che tutte le operazioni di lettura da parte di un membro della famiglia 68000 hanno bisogno di quattro cicli di clock in più delle operazioni di

scrittura; il numero dei cicli di clock necessari dipende infatti dalla fase di E. Notate che E ha un duty cicle (ciclo di lavoro) del 40%: ciò significa che è alto per 4 cicli e basso per sei. Durante il ciclo di scrittura mostrato in precedenza, E è in sincronia con il ciclo relativo all'esecuzione delle istruzioni. In questo caso quindi, il ciclo di scrittura ha bisogno, per essere eseguito, del minor numero possibile di cicli di CLK. Notate che il 68000 inserisce automaticamente degli stati di wait al fine di potersi sincronizzare con E. Quando il 68000 riceve il segnale $\overline{VPA}$, esso manda in output alla periferica il segnale $\overline{VMA}$. Nelle applicazioni pratiche questo segnale viene spesso collegato al chip selettore di input della periferica. Notate che il 68008 non possiede un segnale $\overline{VMA}$: la logica esterna deve quindi attivare il segnale $\overline{VMA}$. Alla fine del ciclo di lettura o di scrittura, la periferica di tipo 6800 o la logica di sistema preposta alla decodifica degli indirizzi deve negare il segnale $\overline{VPA}$ entro la fine del periodo di clock successivo alla negazione da parte del 68000 del segnale $\overline{AS}$. Ciò impedisce al processore di assumere che il ciclo successivo sia un ciclo di tipo 6800.

APPLICABILITÀ

A tutt'oggi la maggior parte delle periferiche necessarie è stata realizzata implementando il trasferimento asincrono: i nuovi sistemi non hanno quindi più bisogno di usare i segnali di interfacciamento con la famiglia 6800. Per questo motivo il 68020 non include questi segnali nel proprio package.

B.2 Interfaccia con il coprocessore

Il 68020 comprende un'interfaccia con il coprocessore. Un coprocessore è un processore dedicato a un compito particolare che agisce come estensione del processore principale. Per il 68020 sono attualmente disponibili due coprocessori: uno è un coprocessore matematico in virgola mobile (68881) e l'altro è un gestore di memoria paginata (68851).
I coprocessori differiscono dalle periferiche ordinarie a causa del loro interfacciamento con processore. Mentre il processore comunica con le altre periferiche attraverso normali istruzioni (ad esempio MOVE), le comunicazioni con il coprocessore avvengono in modo trasparente rispetto al programmatore assembler: il coprocessore appare quindi come parte integrante del processore principale.
Le istruzioni del 68020 per il coprocessore si presentano come istruzioni di linea F (come illustrato sotto). Quando il processore incontra un'istruzione di questo tipo, esso inizia un ciclo sullo spazio di CPU. Come parte di questo ciclo il processore codifica il numero del processore di cui è richiesto l'in-

tervento estraendolo dal codice dell'istruzione. Se il coprocessore non è presente, la logica esterna attiva $\overline{BERR}$ e il processore viene intrappolato nel vettore relativo alla linea F.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	cpid			0	0	0	modalità			registro		
Comando del coprocessore															
Word di estensione (opzionale)															

Se il coprocessore è presente, lui e il processore principale cominciano una serie di tresferimenti bidirezionali. Il processore principale comunica al coprocessore qual è il comando richiesto (come indicato dal codice operativo dell'istruzione); il coprocessore richiede al processore principale di rintracciare tutti gli operandi necessari e quindi esegue il comando. Dopo l'esecuzione del comando, il coprocessore restituisce tutti i dati richiesti al processore principale. Mentre il coprocessore sta eseguendo il comando, il processore principale continua a interrogare il coprocessore fino a che questi non ha terminato il proprio lavoro. A questo punto il processore può continuare la normale esecuzione delle istruzioni.
Notate che il modo in cui avvengono le operazioni non differisce molto dal modo in cui un processore può interfacciarsi a una periferica che non è un coprocessore. La differenza principale sta nel livello di macrocodice e microcodice richiesto: come già detto il processore usa una serie di istruzioni standard per comunicare con le periferiche. Le istruzioni eseguite per interfacciarsi con il coprocessore sono realizzate interamente in microcodice; esse non richiedono alcun fetch di istruzioni esterne e quindi possono essere eseguite molto più velocemente. Dal momento che sono realizzate in microcodice il programmatore non deve conoscere i dettagli dell'interfaccia con il coprocessore, ma solo la sintassi delle sue istruzioni.
Il 68020 è l'unico membro della famiglia 68000 che possiede il microcodice per l'interfacciamento con il coprocessore. In ogni caso, dal momento che l'interfacciamento non richiede l'utilizzo di linee di controllo addizionali, gli altri membri della famiglia 68000 potrebbero ragionevolmente interfacciarsi al coprocessore emulando i trasferimenti nello spazio di CPU effettuati dal 68020. I dettagli di questo interfacciamento vanno al di là degli scopi del presente libro; il coprocessore disponibile per il 68020 opera in modo trasparente all'utente e perciò noi non entreremo nei dettagli relativi alle particolari primitive del coprocessore medesimo.

C

Differenze fra i membri della famiglia 68000

Questa appendice riassume le differenze fra i diversi membri della famiglia 68000.

Dimensione del bus dati (in bit):

68000	16
68008	8
68010	16
68012	16
68020	8, 16, 32

Dimensione del bus indirizzi (in bit):

68000	24
68008	20
68010	24
68012	30 (più A31)
68020	32

Cicli della cache:

68000	Nessuno
68008	Nessuno
68010	Ciclo di 3 word
68012	Ciclo di 3 word
68020	Cache di 128 word

Memoria virtuale/macchina virtuale:

68000	No
68008	No
68010	Sì
68012	Sì
68020	Sì

Allineamento della memoria:

68000	Allineamento pari per le word, le long word, per le istruzioni e per lo stack
68008	Allineamento pari per le word, le long word, per le istruzioni e per lo stack
68010	Allineamento pari per le word, le long word, per le istruzioni e per lo stack
68012	Allineamento pari per le word, le long word, per le istruzioni e per lo stack
68020	Allineamento pari solo per le istruzioni

Registri di controllo:

68000	Nessuno
68008	Nessuno
68010	SFC,DFC,VBR
68012	SFC,DFC,VBR
68020	SFC,DFC,VBR,CACR,CAAR

Stack pointer:

68000	USP,SSP
68008	USP,SSP
68010	USP,SSP
68012	USP,SSP
68020	USP,SSP(MSP,ISP)

Modi di indirizzamento tipici del 68020:

indiretto in memoria, indice scalato, spiazzamenti maggiori

Istruzioni tipiche del 68020

Bcc	Supporta spiazzamenti di 32 bit
BFxx	Nuovo tipo di istruzioni

BKPT	Supporta la sostituzione di operatori esterni
BRA	Supporta spiazzamenti di 32 bit
BSR	Supporta spiazzamenti di 32 bit
CALLM	Nuova istruzione
CAS,CAS2	Nuova istruzione
CHK	Supporta operandi di 32 bit
CHK2	Nuova istruzione
CMPI	Nuove modalità di indirizzamento
CMP2	Nuova istruzione
cpxx	Nuovo tipo di istruzioni
DIVS,DIVU	Supporta operandi di 32 e 64 bit
EXTB	Supporta estensioni da 8 bit fino a 32 bit
LINK	Supporta spiazzamenti di 32 bit
MOVEC	Supporta nuovi registri di controllo
MULS/MULU	Supporta operandi di 32 bit
PACK	Nuova istruzione
RTM	Nuova istruzione
TST	Nuova modalità di indirizzamento
TRAPcc	Nuova istruzione
UNPK	Nuova istruzione

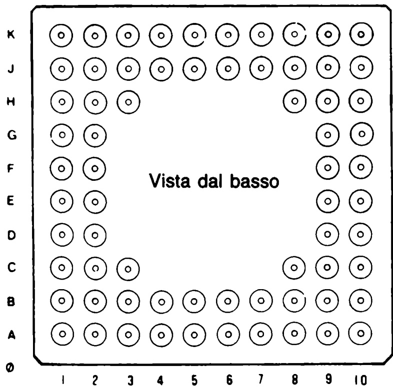
D

Packaging

Chip carrier a 68 terminali (68000, 68010)

Pin Grid Array (PGA) a 114 terminali (68020)

Pin Grid Array (PGA) a 68 terminali (68000, 68010)



Assegnamento dei piedini per un PGA a 68 terminali (68000, 68010)

Numero piedino	Funzione
A1	Do Non connesso
A2	$\overline{AS}$
A3	D1
A4	D2
A5	D4
A6	D5
A7	D7
A8	D8
A9	D10
A10	D12
B1	$\overline{DTACK}$
B2	$\overline{LDS}$
B3	$\overline{UDS}$
B4	D0
B5	D3
B6	D6
B7	D9
B8	D11
B9	D13
B10	D15
C1	$\overline{BGACK}$
C2	$\overline{BG}$
C3	R/$\overline{W}$
C8	D13
C9	A23
C10	A22
D1	$\overline{BR}$
D2	V_{CC}
D9	V_{SS}
D10	A21
E1	CLK
E2	V_{SS}
E9	V_{CC}
E10	A20

Numero piedino	Funzione
F1	$\overline{HALT}$
F2	$\overline{RESET}$
F9	A18
F10	A19
G1	$\overline{VMA}$
G2	$\overline{VPA}$
G9	A15
G10	A17
H1	E
H2	$\overline{IPL2}$
H3	$\overline{IPL1}$
H8	A13
H9	A12
H10	A16
J1	$\overline{BERR}$
J2	$\overline{IPL0}$
J3	FC1
J4	Do Non connesso
J5	A2
J6	A5
J7	A8
J8	A10
J9	A11
J10	A14
K1	Do Non connesso
K2	FC2
K3	FC0
K4	A1
K5	A3
K6	A4
K7	A6
K8	A7
K9	A9
K10	Do Non connesso

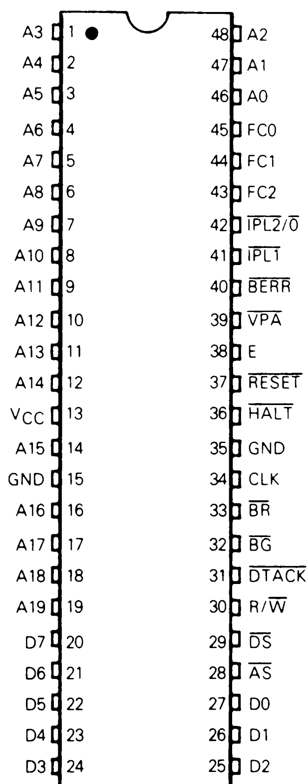
Assegnamento dei piedini per un PGA a 114 terminali (68020)

Numero piedino	Funzione
A1	$\overline{BGACK}$
A2	A1
A3	A31
A4	A28
A5	A26
A6	A23
A7	A22
A8	A19
A9	V_{CC}
A10	GND
A11	A14
A12	A11
A13	A8
B1	N.C.
B2	$\overline{BG}$
B3	$\overline{BR}$
B4	A30
B5	A27
B6	A24
B7	A20
B8	A18
B9	GND
B10	A15
B11	A13
B12	A10
B13	A6
C1	$\overline{RESET}$
C2	CLOCK
C3	N.C.
C4	A0
C5	A29
C6	A25
C7	A21
C8	A17
C9	A16
C10	A12
C11	A9
C12	A7
C13	A5

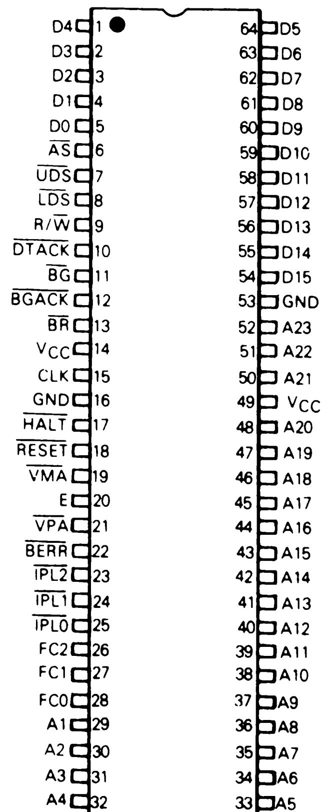
Numero piedino	Funzione
D1	V_{CC}
D2	V_{CC}
D3	N.C.
D4-D11	–
D12	A4
D13	A3
E1	FC0
E2	$\overline{RMC}$
E3	V_{CC}
E4-E11	–
E12	A2
E13	$\overline{OCS}$
F1	SIZ0
F2	FC2
F3	FC1
F4-F11	–
F12	N.C.
F13	$\overline{IPEND}$
G1	$\overline{ECS}$
G2	SIZ1
G3	$\overline{DBEN}$
G4-G10	–
G11	V_{CC}
G12	GND
G13	V_{CC}
H1	$\overline{CDIS}$
H2	$\overline{AVEC}$
H3	$\overline{DSACK0}$
H4-H11	–
H12	IPL2
H13	GND
J1	$\overline{DSACK1}$
J2	$\overline{BERR}$
J3	GND
J4-J11	–
J12	$\overline{IPL0}$
J13	$\overline{IPL1}$

Numero piedino	Funzione
K1	GND
K2	$\overline{HALT}$
K3	N.C.
K4-K11	–
K12	D1
K13	D0
L1	$\overline{AS}$
L2	$R/\overline{W}$
L3	D30
L4	D27
L5	D23
L6	D19
L7	GND
L8	D15
L9	D11
L10	D7
L11	N.C.
L12	D3
L13	D2
M1	$\overline{DS}$
M2	D29
M3	D26
M4	D24
M5	D21
M6	D18
M7	D16
M8	V_{CC}
M9	D/3
M10	D10
M11	D6
M12	D5
M13	D4
N1	D31
N2	D28
N3	D25
N4	D22
N5	D20
N6	D17
N7	GND
N8	V_{CC}
N9	D14
N10	D12
N11	D9
N12	D8
N13	N.C.

Package di tipo dual inline a 48 piedini (68008)



Package di tipo dual inline a 64 piedini (68000, 68010)



(Le informazioni sul packaging del 68012 non sono disponibili)

E

Operazioni con la cache per il 68020

Gran parte del tempo di esecuzione di un processore viene spesa eseguendo cicli. Un modo comune di migliorare le prestazioni del processore è quello di usare una cache, una memoria ad alta velocità, che permetta al processore di accedere molto più velocemente alle istruzioni di cui ha bisogno. Una volta lette le istruzioni nella cache il processore può eseguirle molto più rapidamente rispetto al tempo che sarebbe stato necessario se le avesse dovute rintracciare dalla memoria esterna.
Molte configurazioni non usano alcun stato di wait durante i cicli di memoria: il risultato di ciò è una notevole accelerazione della velocità del processore. Il 68020 si spinge ancora più in là dal momento che implementa una cache istruzioni direttamente sul chip: l'accesso a questa cache può quindi avvenire senza far ricorso a bus esterni o ad altri meccanismi.
La cache sul chip può contenere 64 elementi di tipo long word: ogni elemento consiste di un campo "tag", un bit di validità e 32 bit di dati relativi all'istruzione. Il campo tag è formato dai 24 bit più alti (A8-A31) dell'indirizzo e dal valore FC2 che differenzia lo spazio utente dallo spazio supervisore.
Ogni volta che il 68020 richiede il fetch di un'istruzione, esso controlla che questa sia presente nella cache; ciò viene fatto indicizzando la cache usando i bit A2-A7 dell'indirizzo istruzioni. Esso controlla quindi il campo tag e il codice FC2 associato con questo elemento. Se questi concordano e il bit di validità è settato la cache è stata "colpita". Il bit A1 seleziona l'offset dell'elemento di tipo long word e il processore può dare inizio all'esecuzione delle istruzioni senza effettuare alcun fetch.
Se il campo tag non corrisponde o il bit di validità non è settato si verifica un "fallimento" e il processore effettua il fetch dell'istruzione dalla memoria usando un ciclo di lettura standard. Appena letta, l'istruzione viene inse-

rita nella cache, viene aggiornato il campo tag e settato il bit di validità. Notate che il 68020 usa la cache solo per le istruzioni: il fetch dei dati richiede sempre l'effettuazione di cicli di memoria esterni.

E.1 Controllo della cache

La cache istruzioni non sempre è necessaria e non sempre è utile. Ad esempio durante l'emulazione hardware il sistema potrebbe richiedere l'effettuazione di soli fetch esterni al fine di facilitare alcuni aspetti dell'emulazione. Il 68020 può controllare l'accesso alla cache attraverso il registro di controllo della cache (CACR) e il registro degli indirizzi della cache (CAAR). Il CAAR è usato solo per cancellare gli elementi della cache (vedere CE più sotto). Il CACR è un registro a 32 bit con quattro bit definiti come in Figura E.1. Il processore accede al CACR con l'istruzione MOVEC. Notate che questa istruzione è privilegiata. I bit di CACR sono definiti come segue:

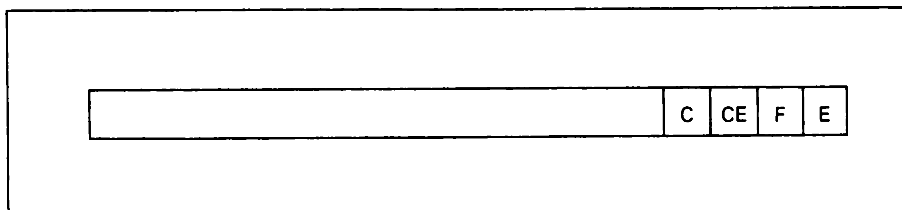


Figura E.1 Registro di controllo della cache (CACR).

E - Abilitazione della cache Questo bit, quando viene settato, permette il normale funzionamento della cache. Quando viene cancellato esso disabilita la cache e fa sì che tutte le fasi di fetch siano rivolte alla memoria esterna. Un reset hardware pulisce sempre questo bit.

F - Congelamento della cache Questo bit, quando viene settato, fissa il contenuto della cache. La cache può ancora venire "colpita" ma i "fallimenti" non provocheranno sostituzioni all'interno della cache stessa. Un reset hardware pulisce sempre questo bit.

CE - Pulizia di un elemento (Clear Entry) Questo bit, quando viene settato, pulisce il bit di validità dell'elemento della cache associato all'indice dato dal CAAR. Questa funzione viene eseguita una sola volta quando il CACR viene scritto con l'istruzione MOVEC (il bit è a sola scrittura). Il formato del CAAR è mostrato in Figura E.2.

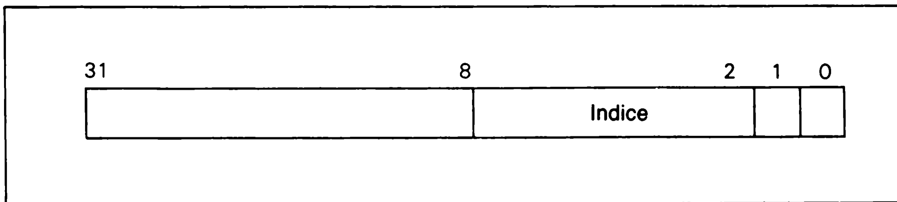


Figura E.2 Registro indirizzi della cache.

C - Pulizia della cache (Clear cache) Questo bit a sola scrittura, quando viene settato, invalida tutti gli elementi della cache (ad esempio come parte di un cambiamento di contesto di un task).

E.2 Disabilitazione della cache

Il segnale di disabilitazione della cache ($\overline{\text{CDIS}}$) disabilita dinamicamente la cache. Dopo che la logica esterna ha attivato $\overline{\text{CDIS}}$, tutti i fetch delle istruzioni successive verranno effettuati nella memoria esterna indipendentemente dal bit di abilitazione della cache del CACR. Quando la logica esterna nega $\overline{\text{CDIS}}$ la cache viene riabilitata per i fetch successivi.

Indice analitico

La McGraw-Hill pubblica in tutto il mondo centinaia di libri di informatica per lo studio, la professione e il tempo libero. La produzione in lingua italiana comprende:

88 386 0001 5	J. Heilborn e R. Talbott, *Guida al Commodore 64*
88 386 0002 3	C.A. Street, *La gestione delle informazioni con lo ZX Spectrum*
88 386 0003 1	T. Woods, *L'Assembler per lo ZX Spectrum*
88 386 0004 X	R. Jeffries, G. Fisher e B. Sawyer, *Divertirsi giocando con il Commodore 64*
88 386 0005 8	G. Bishop, *Progetti hardware con lo ZX Spectrum*
88 386 0006 6	H. Mullish e D. Kruger, *Il BASIC Applesoft*
88 386 0007 4	N. Williams, *Progettazione di giochi d'avventura con lo ZX Spectrum*
88 386 0008 2	H. Peckham, *Il BASIC e il PC-IBM in pratica*
88 386 0009 0	H. Peckham, *Il BASIC e il Commodore 64 in pratica*
88 386 0010 4	S. Nicholls, *Tecniche avanzate in Assembler con lo ZX Spectrum*
88 386 0011 2	K. Skier, *L'Assembler per il Commodore 64 e il VIC-20*
88 386 0012 0	S. Kamins e M. Waite, *Programmazione umanizzata in Applesoft*
88 386 0013 9	A. Pennell, *Guida allo ZX Microdrive e all'Interface 1*
88 386 0015 5	P. Cohen, *Grafica e animazione con gli Apple II*
88 386 0016 3	C. Duff, *Guida al Macintosh*
88 386 0017 1	G. Kane, *Il manuale MC68000*
88 386 0018 X	P. Hoffman e T. Nicoloff, *Il manuale MS-DOS*
88 386 0019 8	E.M. Baras, *Come usare il Symphony*
88 386 0020 1	S. Nicholls, *Grafica avanzata con lo ZX Spectrum*
88 386 0021 X	L.J. Graham e T. Field, *Guida al PC-IBM*
88 386 0022 8	T. Field, *Come usare MacWrite e MacPaint*
88 386 0024 4	H. Peckham, *Il BASIC e gli Apple II in pratica*
88 386 0025 2	C. Morgan e M. Waite, *Il manuale 8086/8088*
88 386 0026 0	W. Ettlin, *Come usare il Multiplan*
88 386 0027 9	G. Mainis, *Il manuale ProDOS*
88 386 0028 7	J. Jones, *Il SuperBASIC del QL*
88 386 0029 5	C. Opie, *L'Assembler per il QL*
88 386 0030 9	W. Ettlin e G. Solberg, *Il BASIC Microsoft*
88 386 0031 7	D.L. Toppen, *Il Forth in pratica*
88 386 0032 5	R. Person, *Le meraviglie dell'animazione con gli Apple II*
88 386 0033 3	P.A. Sand, *Programmazione avanzata in Pascal*
88 386 0034 1	P. Hoffman, *Il manuale MSX*
88 386 0035 X	R. Person, *Le meraviglie dell'animazione con il PC-IBM*
88 386 0036 8	W. Ettlin, *Il Multiplan per il Macintosh*
88 386 0037 6	D. Kruglinski, *Introduzione al Framework*
88 386 0038 4	L. Barnes, *Come usare il dBase II*
88 386 0040 6	L. Barnes, *Come usare il dBase III*
88 386 0041 4	W. Ettlin e G. Solberg, *Il GW-BASIC per Personal Computer Olivetti*
88 386 0042 2	D. Watt, *Il LOGO per il Commodore 64*
88 386 0043 0	T.J. Byers, *Guida al PC AT IBM*
88 386 0044 9	D. Kater e R. Kater, *Guida alle stampanti Epson*
88 386 0045 7	R.L. Tokheim, *Progetti di circuiti elettronici per microcalcolatori*
88 386 0047 3	C. Siechert e C. Wood, *Il manuale MS-DOS 3.20*
88 386 0048 1	H. Peckham, *Programmazione strutturata in BASIC*
88 386 0049 X	W.H. Murray III e C.H. Pappas, *L'Assembler per l'80286/80386*

88 386 0050 3	D. Andersen, C. Cooper e B. Dempsey, *dBase III in pratica*
88 386 0051 1	D.W. Carroll, *Programmazione in Turbo Pascal*
88 386 0052 X	P. Hoffman e T. Nicoloff, *Il manuale MS-DOS per Personal Computer Olivetti*
88 386 0053 8	C.W. Kyd, *Lotus 1-2-3 - Applicazioni finanziarie*
88 386 0054 6	M. Brownstein, *Come usare Microsoft Word*
88 386 0055 4	J. Heilborn, *Guida al Commodore 128*
88 386 0056 2	E. Jones, *Come usare dBase III Plus*
88 386 0058 9	W. Cramer e G. Kane, *Il manuale MC68000*
88 386 0060 0	P. Robinson, *Programmazione in Turbo Prolog*
88 386 0063 5	A.E. Stanley, *Il BASIC 7.0 per il Commodore 128*
88 386 0064 3	W.A. Ettlin, *Come usare il WordStar 4.0*
88 386 0065 1	M. Campbell, *Guida completa LOTUS 1-2-3*
88 386 0066 X	J.D. Carrabis, *Guida completa dBASE III PLUS*
88 386 0069 4	H. Schildt, *Turbo Prolog 1.1 - Programmazione avanzata*
88 386 0071 6	H. Schildt, *Turbo C - Elementi di programmazione*
88 386 0073 2	H. Schildt, *Turbo C - Programmazione avanzata*
88 386 0075 9	H. Schildt, *Turbo Pascal - Programmazione avanzata*
88 386 0076 7	S. Wood, *Guida al Turbo Pascal 4.0*
88 386 0081 3	K. Jamsa, *Guida completa DOS*
88 386 0089 9	M. Liskin, *Guida tascabile - dBase III Plus*
88 386 0094 5	E. Iacobucci, *OS/2 Manuale di programmazione*
88 386 0098 8	M. Campbell, *Guida tascabile - Lotus 1-2-3*
88 386 0100 3	H. Schildt, *Guida tascabile - C*
88 386 0102 X	K. Jamsa, *Guida tascabile - MS-DOS*
88 386 0103 8	K. Jamsa, *Guida tascabile - OS/2*
88 386 0104 6	K. Jamsa, *Guida tascabile - Turbo Pascal 4.0*
88 386 0106 2	S. Cobb, *Guida tascabile - Quattro*
88 386 0107 0	G. Jones, *Guida tascabile - Paradox*
88 386 0137 2	C. Siechert e C. Wood, *Il manuale MS-DOS 3.20 per PC 1 Olivetti PRODEST*
88 386 0138 0	W. Ettlin e G. Solberg, *Il GW-BASIC per PC 1 Olivetti PRODEST*
88 386 0143 7	H. Schildt, *Turbo Pascal 4.0 - Programmazione avanzata*
88 386 0146 1	M. Guerriero e H. Zampariolo, *Programmare in FRED con Framework II e III*
88 386 7008 0	D.W. Carroll, *Programmazione in Turbo Pascal* (edizione scolastica)
88 386 7009 9	M.L. Schagrin, W.J. Rapaport e R.D. Dipert, *Logica e computer* (edizione scolastica)
88 386 0601 3	S. Harrington, *Computer Graphics - Corso di programmazione*
88 386 0602 1	O. Lecarme e J.L. Nebut, *Pascal - Guida per programmatori*
88 386 0603 X	M. McGilton e R. Morgan, *Il sistema operativo UNIX*
88 386 0604 8	E. Rich, *Intelligenza Artificiale*
88 386 0605 6	M.L. Schagrin, W.J. Rapaport e R.D. Dipert, *Logica e computer*
88 386 0606 4	W. Newman e R. Sproull, *Principi di Computer Graphics*
88 386 0607 2	L. Hancock e M. Krieger, *Il linguaggio C*
88 386 0608 0	R. Thomas, L.R. Rogers e J.L. Yates, *UNIX System V Complementi di programmazione*
88 386 0609 9	S. Chapra e R. Canale, *Metodi numerici per l'ingegneria*
88 386 0610 2	P.R. Gray e R.G. Meyer, *Circuiti integrati analogici - Analisi e progetto*
88 386 0613 7	R. Morgan e H. McGilton, *Il sistema operativo UNIX System V*
88 386 0614 5	A.M. Mood, F.A. Graybill, D.C. Boes, *Introduzione alla statistica*

88 386 0615 3	J.W.L. Ogilvie, *Il linguaggio Modula-2*
88 386 0616 1	R. Holloway, *Progettazione di strutture con i microcalcolatori*
88 386 0619 6	F. Ramade, *Catastrofi ecologiche*
88 386 0636 6	D. Mandrioli, *Elementi di informatica*

La produzione software comprende:

88 386 0902 0	C.A. Street, *PROFILE 2 - Foglio elettronico integrato per lo ZX Spectrum*
88 386 0903 9	S. Nicholls, *Routines in Assembler per la grafica avanzata con lo ZX Spectrum*
88 386 0904 7	A. Bleasby, *Assembler/Disassembler per il Commodore 64*
88 386 0905 5	ACS Software, *ZX Spectrum Monitor*
88 386 0906 3	G. Fitzgibbon, *Projector 1 - Uno strumento per costruire presentazioni grafiche con lo ZX Spectrum*
88 386 0907 1	C. Opie, *QL Machine Code Editor/Assembler*
88 386 0908 X	B. Thompson e W. Thompson, *Sistema Esperto McGraw-Hill Versione 1.0 per personal MS-DOS*
88 386 0909 8	A. Tal, *Generatore di lezioni per il Commodore 64*
88 386 0911 X	B. Thompson e W. Thompson, *Sistema Esperto McGraw-Hill per Apple II*
88 386 0912 8	D. Carroll, *Programmazione in Turbo Pascal*
88 386 0917 6	P. Lazzarini e G. Sarnataro, *CARTESIO-Microlinguaggio per lo studio delle trasformazioni geometriche*
88 386 0918 7	B. Thompson e W. Thompson, *Sistema Esperto McGraw-Hill Versione 2.0 per personal MS-DOS*
88 386 0921 7	G. Montuori e S. Raggi, *Diario di classe*
88 386 0922 5	G. Montuori e S. Raggi, *Registro di classe*
88 386 0923 3	A. Catalani, D. Biolghini, G. Montuori, *Basi di conoscenza di Matematica per il Sistema Esperto McGraw-Hill*

Questo volume, sprovvisto del talloncino a fronte, è da considerarsi copia saggio e campione gratuito fuori commercio. Fuori campo applicazione IVA ed esente da bolla di accompagnamento (art. 22 L. 67/1987, art. 2 lett. i DPR 633/1972 e art. 4 n. 6 DPR 627/1978).

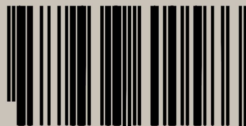
Lire 30 000

Cramer-Kane
Il manuale MC68000
McGraw-Hill Libri Italia
88 386 0058 9

La famiglia dei microprocessori Motorola 68000 costituisce un riconosciuto punto di riferimento per tutti i progettisti di sistemi avanzati. Le notevoli caratteristiche in termini di velocità di esecuzione, programmabilità e indirizzamento di memoria, fanno di questi microprocessori, in particolare il 68020, la CPU ideale per workstation, sistemi CAD/CAM, robot e controllori di processo.
Questo manuale descrive in dettaglio le caratteristiche funzionali dei microprocessori e le possibili architetture di sistemi realizzabili attorno a essi.
Tra gli argomenti trattati:

- descrizione delle modalità di funzionamento
- indirizzi e modalità di indirizzamento
- set di istruzioni
- descrizione dei segnali
- temporizzazione e gestione dei bus
- operazioni di reset
- gestione delle exception
- interfacce
- packaging.

ISBN 88-386-0058-9



Lire 30 000

W. Cramer
G. Kane

Il manuale MC68000

058-9

McGraw-Hill